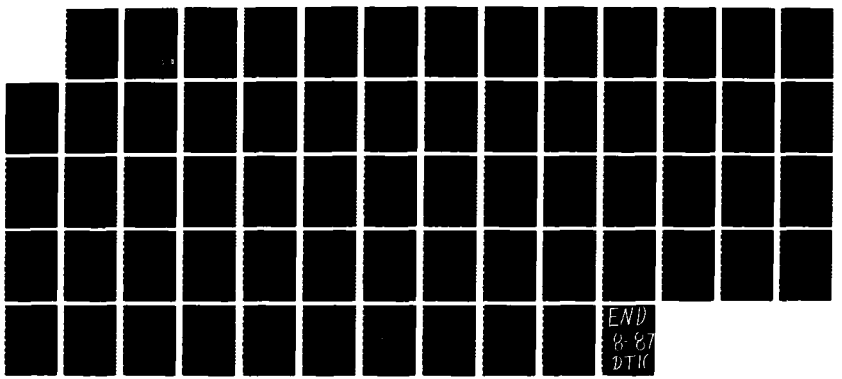
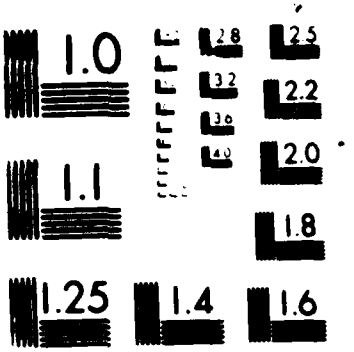
AD-A182 868 AN EFFICIENT WAY FOR EDGE-CONNECTIVITY AUGMENTATION(U) 1/1
ILLINOIS UNIV AT URBANA APPLIED COMPUTATION THEORY
GROUP T WATANABE APR 87 ACT-76 N00014-84-C-0149

UNCLASSIFIED F/G 12/2 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8

April 1987

UILU-ENG-87-2221
ACT-76

COORDINATED SCIENCE LABORATORY
College of Engineering
Applied Computation Theory

AN EFFICIENT WAY FOR EDGE-CONNECTIVITY AUGMENTATION

Toshimasa Watanabe

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A182 368

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2221 (ACT-76)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Joint Services Electronics Program 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-84-C-0149
8c. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
AN EFFICIENT WAY FOR EDGE-CONNECTIVITY AUGMENTATION

12. PERSONAL AUTHOR(S)
Watanabe, Toshimasa

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	April 1987	62

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	edge-connectivity augmentation problem, algorithm, computational complexity

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

We present an algorithm for finding a minimum set of edges to be added so as to k-edge-connect a given graph $G = (V,E)$ with $k > 1$ and $|V| > 1$. The time complexity is $O(k^2|V|^3(k|V| + |E|))$ or $O(k^2(|V|^4 + k|V| + |E|))$ if we use Dinic's maximum flow algorithm or Malhotra, Kumar and Maheshwari's one, respectively, as a subroutine.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

AN EFFICIENT WAY FOR EDGE-CONNECTIVITY AUGMENTATION

Toshimasa Watanabe*

Keywords: Edge-connectivity augmentation problem; algorithm; computational complexity

* Visiting Associate Professor, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801. Associate Professor, Department of Applied Mathematics, Faculty of Engineering, Hiroshima University, Saijo, Higashi Hiroshima 724, JAPAN.

ABSTRACT

We present an algorithm for finding a minimum set of edges to be added so as to k-edge-connect a given graph $G = (V,E)$ with $k > 1$ and $|V| > 1$. The time complexity is $O(k^2|V|^3(k|V| + |E|))$ or $O(k^2(|V|^4 + k|V| + |E|))$ if we use Dinic's maximum flow algorithm or Malhotra, Kumar and Maheshwari's one, respectively, as a subroutine.

1. INTRODUCTION

The problem in which the object is to add a minimum weight set of edges to a graph $G = (V,E)$ so as to satisfy a given vertex- or edge-connectivity condition is called the *vertex- or edge-connectivity augmentation problem*. This problem has a wide variety [3,4,5,10,14-22].

Frank and Chou [5] discussed the unweighted version of some edge-connectivity augmentation problem for graphs without edges, and showed that it is polynomially solvable. Eswaran and Tarjan [3] considered the following problems:

(i) The strong connectivity augmentation problem for directed graphs.

(ii) The bridge-connectivity augmentation problem for undirected graphs.

(iii) The biconnectivity augmentation problem for undirected graphs.

They proved that the weighed versions of these three types are NP-complete and that each of the unweighted versions has an $O(|V| + |E|)$ algorithm. Rosenthal and Goldner [15] proposed an $O(|V| + |E|)$ algorithm for the unweighted version of the biconnectivity augmentation problem. Frederickson and Ja'ja' [6] discussed the NP-completeness of several restricted augmentation problems and showed $O(|V|^2)$ approximation algorithms for above problems (i)-(iii).

We are interested in the k-edge connectivity augmentation problem for undirected graphs with $k \geq 2$, a generalization of (ii). The weighted version of the problem are easily shown to be NP-complete. The unweighted version, which had been one of open problems in graph theory [2, p. 49], was solved by Watanabe and Nakamura [20-22]: it is shown that the cardinality of a minimum solution of the problem is equal to the k-augmentation number, $EA_k(G)$, of a given graph G (the definition will be given later) and that a minimum solution can be obtained in $O(k^2|V|^4(k|V| + |E|)$ time by using Dinic's maximum flow algorithm.

In this paper we consider an improvement of our previous algorithm, given in [20-22], for k-edge-connectivity augmentation problems.

In section 2, graph-theory terminologies and technical terms used in this paper are given.

In section 3, we summarize our previous results on k-edge-connectivity augmentation problems.

In section 4, we describe an improvement of the algorithm mentioned in [20-22]. The previous algorithm repeats two procedures: the one constructs the data structure called the component tree by using a maximum flow algorithm, and the other searches for a pair of vertices, called an admissible pair, which are to be joined by a new edge. The most time-consuming part of the algorithm is the first procedure, which is repeated each time a new edge is added. Thus it is repeated $EA_k(G)$ times, where $EA_k(G) \leqslant k|V|$.

The definition of an admissible pair implies that any minimum solution Z of the problem is partitioned into some minimal sets $Z(m+1),\ldots,Z(k)$ such that their addition in this order increases the edge connectivity one by one, where m is equal to the edge connectivity of G.

We will describe how to determine such a minimal set of new edges whose addition to the current graph increases the edge connectivity by exactly one, without reconstructing the data structure. This will reduce the repetition of reconstructing the data structure to at most k-1 times, leading to a more efficient algorithm.

We consider the case where a given graph is disconnected or connected, respectively, in 4.1 or 4.2. In 4.3, we estimate the time complexity of the improved algorithm and show that it is

$$O(k^2|V|^3(k|V| + |E|))$$

if we use Dinic's maximum flow algorithm [4] or

$$O(k^2(|V|^4 + k|V| + |E|))$$

if we use the maximum flow algorithm proposed by Malhotra, Kumar and Maheshwari [4.13].

2. PRELIMINARIES

Many of graph-theory terminologies and technical terms used in this paper are more or less standard, and those not specified here can be identified in [1,4,8].

A *graph* $G = (V,E)$ (or $G=(V(G),E(G))$) is a finite set of vertices, V, and a finite set of edges, E. If E is a multiset, that is, if any edge may occur several times, then G is called a *multigraph*. Such edges are called multiple edges. Otherwise G is a *simple graph*. In this paper, the term "a graph" means an undirected multigraph unless otherwise stated.

Two vertices u,v which comprise an edge are said to be *adjacent*, and the edge is often denoted by (u,v), even if it is one of multiple edges, as long as no confusion arises. The edge (u,v) is incident to the vertices u,v: u and v are incident to (u,v). The *degree* $d_G(v)$ (or, simply d (v)) of a vertex v of G is the number of edges incident to it in G. An edge (v,v), that is, an edge joining v to itself is referred to as a *loop*. $G_1 = (V_1,E_1)$ is *isomorphic* to $G_2 = (V_2,E_2)$ if $|V_1| = |V_2|, |E_1| = |E_2|$ and there is a bijection ξ of V_1 onto V_2 such that $(u,v)\in E_1$ if and only if $(\zeta(u),\zeta(v))\in E_2$.

A *walk* of G from v_1 to v_n (or a (v_1, v_n) - *walk* of G) is an alternating sequence of vertices and edges of G, $v_1,e_1,v_2,e_2,\ldots,v_{n-1},e_{n-1},v_n (n\geq 1)$, such that $e_i = (v_i,v_{i+1}), 1\leq i\leq n-1$. The length of this walk is n-1. A *path* (A *trail*, respectively) is a walk without any repeated vertices (edges) in it. For $1\leq i<j\leq n$, the (v_i,v_j) -path consisting of edges $(v_i,v_{i+1}),\ldots,(v_{j-1},v_j)$ is referred to as the (v_i,v_j)-*subpath* of a (v_1,v_n)-path. If $n > 2$ then $v_2, \ldots, v_{n-1}$ are called the *inner vertices* of the path. If two paths have no edge in common, then they are said to be *edge-disjoint* (or simply, *disjoint*). Let $M_G(u,v)$ (or simply, $M(u,v)$) denote the maximum number of pairwise edge-disjoint (u,v)-paths of G.

G is *connected* if and only if every pair of vertices of G are joined by a path of G. If G and H are two graphs such that $V(H)\subseteq V(G)$ and $E(H)\subseteq E(G)$, then H is a *subgraph* of G. If H is a maximal connected subgraph of G (that is, if $V(H) \neq V(G)$ then G is not connected) then H is called a *connected component* (or simply, *component*) of G. Let Z be a set of edges such that $Z \subseteq E(G)$ ($Z \cap E(G) = \phi$(empty), respectively), where any edge of Z joins two vertices of V(G). Then G-Z

(G+Z, respectively) denotes the graph obtained by deleting all edges of Z from G (by adding all edges of Z to G). If Z = {e} then it is denoted by G-e (G+e) for simplicity.

For two subsets $S,S' \subseteq V(G)$, let $E(S,S';G)$ denote the set of all those edges of E(G) joining a vertex of S and one of S'. In particular, we denote $E(S,V(G)-S;G)$ by $K(S,G)$. If $S = \{v\}$ then we write $K(v,G)$. If $S \neq \phi$ and $S \subset V(G)$ (a proper subset) then K(S,G) is called a *separator* or a $|K(S,G)|$-separator of G. Clearly, if $|V(G)| > 1$ then G-K(S,G) is disconnected. Put

$$d(S,G) = |K(S,G)|,$$

and we call it the *degree* of S (in G).

Let K be a separator of G, and suppose that $K=K(T,G)$ for a nonempty subset $T \subset V(G)$. A pair of disjoint subsets $S, S' \subseteq V(G)$ (that is, $S \cap S' = \phi$) is said to *be separated by* K (or we say that K *separates* S from S') if $S \subseteq T$ and $S' \subseteq V(G)-T$. K is referred to as an *(S, S')-separator* (of G). If $S=\{u\}$ and $S' = \{v\}$ then we simply call K a (u,v)-separator. An (S,S')-separator K with the minimum cardinality among all (S,S')-separators of G is referred to as an *(S,S')-cut*. A (u,v)-cut is defined similarly. Each component of G-K is called a *K-block* (of G). A K-block whose vertex set includes a subset $S \subset V(G)$ is denoted by B(S,K;G) and is referred to as the *(S,K;G)-block* , or simply the *(S,K)-block* of G. (For simplicity, we often use the term "a K-block", meaning its vertex set. If $S=\{u\}$ then $B(\{u\},K;G)$ is written by B(u,K;G).

Let $m \leq k$ for a fixed integer $k > 1$. A subset $S \subseteq V(G)$ is called an *m-edge-component* (or, simply, an *m-component*) of G if and only if the following (1), (2) hold:

(1) $M_G(u,v) \geq m$ for any $u, v \in S$.

(2) For any $u' \in V(G)-S$, S has a vertex v' with $M_G(u',v') < m$.

An m-edge-component that is not an (m+1)-edge-component is said to be *critical*. If S is an m-edge-component of G with $0 \leq d(S,G) < m$ then S is called an *m-pendant* . Clearly, a 1-pendant of G is identical to the vertex set of a component of G. Let $P^m(G)$ denote the total number of m-pendants of G. An m-pendant S of G is referred to as an *external m-pendant* if K(S,G) is an (S,S')-cut of G for some m-edge-component $S' (\neq S)$ of G.

The *edge-connectivity* ec(G) of a graph G is the minimum number of edges whose removal from G disconnect it to more than one component or result in a single vertex:

$$ec(G) = \begin{cases} \min\{\,|K| : K \text{ is a separator of } G\,\} & \text{if } |V(G)| > 1 \\ 0 & \text{otherwise} \end{cases}$$

G is said to be *h-edge-connected* if $ec(G) \geq h$. Let $N_G(u,v)$ (or simply, $N(u,v)$) denote the cardinality of a (u,v)-cut. It is well known that

$$N_G(u,v) = M_G(u,v) \quad \text{for any } u,v \in V(G), u \neq v$$

and that

$$ec(G) = \min\{M_G(u,v): u,v \in V(G),\ u \neq v\} \text{ if } |V(G)| > 1.$$

(See [4.8].)

Let $\lfloor x \rfloor$ ($\lceil x \rceil$, respectively) denote the minimum integer not less than x (the maximum integer not greater than x).

For a subset $S \subseteq V(G)$, let G[S] denote the graph defined by $V(G[S]) = S$ and $E(G[S]) = \{(u,v) \in E(G): u,v \in S\}$. G[S] is referred to as the *subgraph induced by* S of G.

Let Y be a nonempty subset of V(G), and let $a \in Y$. Put

$$G-Y=G[V(G)-Y],$$

and let $G<a,Y>$ be defined as follows:

$$V(G<a,Y>) = (V(G)-Y) \bigcup \{a\},$$

$$E(G<a,Y>) = E(G-Y) \bigcup \{(a,v): (u,v) \in E(G), u \in Y, v \in V(G)-Y\}.$$

It is said in [12] that $G<a,Y>$ *arises from G by identification of Y to a*. Let

$$T(a,a';G) = \{X \subseteq V(G): a \in X,\ a' \in V(G)-X,\ d(X,G) = M_G(a,a')\}.$$

If $a \neq a'$ then $T(a,a';G)$ is nonempty.

THEOREM 2.1. [12].

In a graph G, let $Y \in T(a,a';G)$ for certain $a,a' \in V(G)$. Then, for any distinct vertices u,v of $G<a,Y>$,

$$M_{G<a,Y>}(u,v) = M_G(u,v).$$

Let $S \subset V(G)$, and let G/S denote the graph defined by the following:

$$V(G/S) = V(G) - S \bigcup \{v(S)\} \quad (v(S) \notin V(G)),$$

and

$$E(G/S) = E(V(G) - S, V(G) - S; G) \bigcup \{(u,v(S)): (u,v) \in K(S,G), v \in S\},$$

where $v(s)$ is the new vertex corresponding to S. This operation constructing G/S from G is called *shrinking of S* in G. G/S is called the graph obtained by shrinking of S in G. For simplicity, we also call G/S shrinking of S if no confusion arises. Let $\pi = \{S_1, \ldots, S_t\}$ $(t \geq 2)$ be a partition of $V(G)$:

$$V(G) = S_1 \bigcup \cdots \bigcup S_t, \quad S_i \bigcap S_j = \phi (i \neq j).$$

Let $|\pi|$ denote the total number of sets in π. For each i, $1 \leq i \leq t$, put

$$G_{(i)} = G_{(i-1)} / S_i,$$

where $G_{(0)} = G$. Put

$$G/\pi = G_{(t)},$$

and we call it *the π - shrinking of G*. G/π is uniquely determined up to isomorphism, independently of the order of shrinking of the sets in π. Since we identify two isomorphic graphs, we consider that G/π is unique for each π.

Let $\pi_G(m)$ denote the partition of $V(G)$ into m-components of G, where $m > ec(G)$. Put

$$G/m = G/\pi_G(m)$$

for simplicity. G/m may have multiple edges. Let ρ_m denote a mapping

$$\rho_m: V(G) \to V(G/m) = \{v(S_i): S_i \in \pi(m)\}$$

defined by $\rho_m(v) = v(S_i) \in \pi(m)$ if and only if $v \in S_i$. ρ_m is called the *mapping of G induced by* $\pi_G(m)$. We fix ρ_m and denote $\rho_m^{-1}(v(S_i)) = S_i$. For any set $E \subseteq E(G)$, let

$$\rho_m(E) = \{(\rho_m(u), \rho_m(v)): (u,v) \in E\}.$$

PROPOSITION 2.1.

Let $S_1,S_2 \in \pi(m)(S_1 \neq S_2)$ and $v_1,v_2 \in V(G/m)(v_1 \neq v_2)$. If K_G is an (S_1,S_2)-cut of G then $\rho_m(K_G)$ is a $(\rho_m(S_1),\rho_m(S_2))$-cut of G/m with $|\rho_m(K_G)| = |K_G|$. Conversely, if K is a (v_1,v_2)-cut of G/m then G has a $(\rho_m^{-1}(v_1),\rho_m^{-1}(v_2))$-cut K_G with $|K_G| = |K|$.

PROOF.

It suffices to consider the case where $K_G \neq \phi, K \neq \phi$. Then $m > 1$. Let

$$B_i = \{\rho_m(S):\ S \in \pi(m), S \subset B(S_i,K_G;G)\}, i = 1, 2.$$

It is easy to see that

$$\rho_m(K_G) = E(B_1,B_2;G/m),\ |\rho_m(K_G)| = |K_G|.$$

If we have $S, S' \in B(S_i,K_G;G)\ (S \neq S')$ for either $i = 1$ or $i = 2$ then G has a (w,w')-path P for some pair $w \in S, w' \in S'$ such that

$$E(P) \bigcap K_G^* = \phi.$$

This implies that G/m has a $(\rho_m(S),\rho_m(S'))$-path Q such that

$$E(Q) \bigcap \rho_m(K_G) = \phi.$$

Hence it follows that $\rho_m(K_G)$ is a $(\rho_m(S_1),\rho_m(S_2))$-cut of G/m.

Conversely we prove the second part. For each i, $i = 1, 2$, put

$$B(i) = B(v_i, K; G/m),$$

$$v_i = \bigcup_{v \in B(i)} \rho_m^{-1}(v).$$

Let

$$e_j = (v_{1j},v_{2j}) \in K,\ j = 1, \ldots, m'\ (m' = |K| < m)$$

where

$$v_{ij} \in B(i),\ i = 1, 2.$$

Then, for each j, $1 \leq j \leq m'$, there is an edge

$$f_j = (u_{j1}, u_{j2}) \in E(G),\ u_{jt} \in \rho_m^{-1}(v_{jt}),\ t = 1, 2.$$

Put

$$K_G = \{f_1, \ldots, f_{m'}\}.$$

Then, clearly, we have

$$K_G = E(V_1, V_2; G).$$

Similarly to the proof of the first part we can show that K_G is a $(\rho_m^{-1}(v_1), \rho_m^{-1}(v_2))$-cut of G.

Q.E.D.

Suppose that there is a pair $u,v \in V(G)(u \neq v)$ such that G has exactly p edges

$$e_1, \ldots, e_p \quad (p \geq 1)$$

connecting them. Delete these p edges from G and add an edge e

$$e = (u,v)$$

with weight

$$c(e) = p.$$

We denote

$$\sigma(e_i) = e,\ i = 1, \ldots, p.$$

Repeat such replacement until we obtain a simple graph G^s with each edge having weight equal to the number of multiple edges of G connecting each pair of endvertices.

Construct a directed graph $N(G^s)$ from G^s by replacing each edge $e = (u,v) \in E(G^s)$ by two directed edges

$$e' = (u \rightarrow v),\ e'' = (v \rightarrow u)$$

both of which have weights equal to $c(e)$

$$c(e') = c(e'') = c(e).$$

$N(G^s)$ is called the *network* of G.

Choose a pair of vertices $s, t\ (s \neq t)$ from $V(N(G^s))$ with $M_G(s,t) > 0$, and we call s and t a *source* and a *sink*, respectively. Consider each $c(e)$ to be the capacity of $e \in E(N(G^s))$, and any

existing algorithm for finding a maximum flow f_m from a source to a sink can be applied to $N(G^s)$. Let *val* (f) denote the total flow of a flow f:

$$val(f) = \sum_{e \in IN(t)} f(e) - \sum_{e \in OUT(t)} f(e),$$

where IN(w) (OUT(w), respectively) denotes the set of all edges of $N(G^s)$ incoming to w (outgoing from w). In particular, put

$$F(s \rightarrow t; G) = val(f_m) \text{ or } F(s \rightarrow t) = val(f_m)$$

Let $S, S' \subset V(N(G^s))$ be nonempty disjoint sets, and let

$$E(S \rightarrow S') = \{e = (u \rightarrow v): e \in E(N(G^s)), u \in S, v \in S'\},$$
$$(S,S') = E(S \rightarrow S') \bigcup E(S' \rightarrow S),$$
$$(S,\bar{S}) = (S, V(N(G^s)) - S).$$

We call $(S, \bar{S})$ a cut of $N(G^s)$. Put

$$c(S,\bar{S}) = \sum_{e \in E(S \rightarrow \bar{S})} c(e),$$

which is called the *capacity* of a cut $(S, \bar{S})$. A *minimum cut* is a cut with the minimum capacity among all cuts of $N(G^s)$.

PROPOSITION 2.2.

$$F(s \rightarrow t) = M_G(s,t) \text{ if } M_G(s,t) > 0.$$

PROOF.

Let K be any (s,t)-cut of G, where $|K| = M_G(s,t)$. Put

$$S = B(s,K;G), \quad \bar{S} = B(t,K;G).$$

Let

$$K' = \{\sigma(e) \in E(G^s): e \in K\}.$$

Then clearly,

$$\sum_{\sigma(e)\in K'} c(\sigma(e)) = |K|$$

Hence, for the cut $(S, \bar{S}) \subset E(N(G^s))$ of $N(G^s)$, we have

$$(S,\bar{S}) = E(S\to\bar{S}) \bigcup E(\bar{S}\to S)$$

$$E(S\to\bar{S}) = \{(u\to v) : (u,v)\in K'\},$$

$$E(\bar{S}\to S) = \{(v\to u) : (u,v)\in K'\}$$

Then

$$c(K'') = \sum_{e\in E(S\to\bar{S})} c(e) = |K|, \quad K'' = (S,\bar{S})$$

Since any flow f has

$$val(f) \le c(K''),$$

we have

$$F(s\to t) \le c(K'') = |K| = M_G(s,t).$$

Conversely suppose that we have a maximum flow f_m. Then it is well-known that $N(G^s)$ has a minimum cut $K'' = (S,\bar{S})$ such that

$$s\in S,\ t\in\bar{S},\ c(K'') = F(s\to t)$$

$$f(e) = \begin{cases} c(e) & \text{if } e\in E(S\to\bar{S}) \\ 0 & \text{if } e\in E(\bar{S}\to S) \end{cases}$$

for any $e\in K''$. For each $e = (u\to v)\in E(S\to\bar{S})$ there is $e' = (v\to u)\in E(\bar{S}\to S)$ and vice versa. Let

$$K' = \{(u,v)\in E(G) : (u\to v)\in E(S\to\bar{S})\}$$

Then K' is an (s,t)-separator of G, and

$$\sum_{e \in K''} c(e) = c(K'') = F(s \rightarrow t).$$

Let

$$K = \{e \in E(G): \sigma(e) \in K'\}.$$

Then K is an (s,t)-separator of G with

$$|K| = F(s,t).$$

We have

$$F(s,t) \geq M_G(s,t),$$

since

$$M_G(s,t) = N_G(s,t) \leq |K|.$$

Q.E.D.

COROLLARY 2.1.

For any pair $u,v \in V(G)$ $(u \neq v)$

$$M_G(u,v) = F(u \rightarrow v),$$

where we put $F(u \rightarrow v) = 0$ if $M_G(u,v) = 0$.

3. THE K-EDGE-AUGMENTATION PROBLEM

The *k-edge-connectivity augmentation problem* for any fixed $k \geq 1$ is defined by:

"Given a graph $G = (V, E)$ with $|V| > 1$, determine a minimum set Z of edges joining two vertices in $V(G)$ such that $Z \cap E = \phi$ and $G + Z$ is k-edge-connected."

We can assume that G has no loop and that any added edge joins distinct vertices of V(G). Suppose that $k \geq ec(G)$. Let $R_k(G)$ denote the minimum number of edges whose addition to G result in a k-edge-connected graph. If $k = 1$ then the problem is easy to solve. Therefore we assume that

$$k \geq 2 \text{ and } |V(G)| > 1$$

in this paper.

3.1. THE DEMAND AND THE EDGE-AUGMENTATION NUMBER OF A GRAPH

We give definitions of edge demands of, demands of, component demands of m-components as well as the demand of a graph.

Assume that $ec(G) \leq m \leq k$. Let S denote a nonempty subset of V(G). The *edge demand* of S (of G), $ED_k(S,G)$, is defined by

$$ED_k(S,G) = \begin{cases} 0 & \text{if } S=V(G) \text{ or } d(S,G) \geq k \\ k-d(S,G) & \text{otherwise} \end{cases}$$

Here we denote a t-edge-component of G by S(t) for $t > 0$. If $S = S(m)$ then the demand of S (of G), $D_k(S,G)$, is defined recursively by the following (i), (ii):

(i) If $S = S(k)$ then

$$D_k(S,G) = ED_k(S,G).$$

(ii) If $S = S(m)$ with $m < k$ then

$$D_k(S,G) = \begin{cases} \max(ED_k(S,G), \sum_{S(m+1) \subset S} D_k(S(m+1),G)) & \text{if there is an } S(m+1) \subset S \\ D_k(S(m+1),G) & \text{if } S=S(m+1), \end{cases}$$

where $\sum_{S(m+1) \subset S} D_k(S(m+1),G)$ denotes the total sum of demands $D_k(S(m+1),G)$ of those (m+1)-components $S(m+1) \subset S$ of G.

(We note, as is in Corollary 3.1 of [20-22], that S is the disjoint union of some (m+1)-components of G if S is a critical m-component of G.)

We generalize the definition of $D_k(S,G)$ to that for a subset $S \subseteq V(G)$. Suppose that $S \subseteq V(G)$ and $S \neq S(m)$ for any $m \leq k$, and let

$$h(S) = \begin{cases} \min\{j:\ S(j) \subset S\} & \text{if there is } S(j) \subset S \text{ with } j>0 \\ 0 & \text{otherwise.} \end{cases}$$

Then we define the *component demand* of S (of G), $CD_k(S,G)$, by the following:

$$CD_k(S,G) = \begin{cases} \sum_{S(h(S)) \subset S} D_k(S(h(S)),G) & \text{if } h(S)>0 \\ 0 & \text{otherwise.} \end{cases}$$

For any $S \subseteq V(G)$, the *demand* of S (of G), $D_k(S,G)$, is defined by using this notation as follows:

$$D_k(S,G) = \begin{cases} D_k(S(m),G) & \text{if } S = S(m) \text{ for some } m \leqslant k \\ \max(ED_k(S,G),CD_k(S,G)) & \text{otherwise.} \end{cases}$$

Let $D_k(G)$, called the *demand* of G, denote the value determined by the following procedures (1) - (3):

(1) Compute the demand of S(k), $D_k(S(k),G)$, for every k-component S(k) of G.

(2) If $k \geqslant ec(G)+1$ then, for each m with m=k-1,....,ec(G) in this order, compute recursively the demand of every m-component S(m) of G:

$$D_k(S(m),G) = \begin{cases} \max(ED_k(S(m),G),CD_k(S(m),G)) & \text{if there is } S(m+1) \subset S(m) \\ D_k(S(m+1),G) & \text{if } S(m)=S(m+1). \end{cases}$$

(3) Let

$$D_k(G)=D_k(V(G),G).$$

Put

$$EA_k(G) = \lceil D_k(G)/2 \rceil.$$

We call $EA_k(G)$ the *k-edge-connectivity-augmentation number* (or simply, the *k-augmentation*

number) of G.

3.2. THE CHARACTERIZATION OF $EA_k(G)$.

We summarize our previous results given in [20-22].

PROPOSITION 3.1. [20-22]

For any fixed $k \geq 2$, $G=(V,E)$ is k-edge-connected if and only if $EA_k(G)=0$.

LEMMA 3.1. [20-22]

$R_k(G) \geq EA_k(G)$ for any fixed $k \geq 2$.

We give the definitions of the edge condition and m-augmenting sets, which are necessary to prove the converse of Lemma 3.1:

$$R_k(G) \leq EA_k(G) \text{ for any fixed } k > 1.$$

Clearly, it suffices to consider the case with $ec(G)<k$. For any vertex v and each m with $ec(G) \leq m \leq k$, G has exactly one m-component that contains v. Let $S(v,m;G)$ denote the m-component of G that contains v.

Whenever $ec(G)<k$ we choose from $V(G)$ distinct vertices u_1,u_2 satisfying the following conditions (1) - (3) called the *edge condition* (for G):

(1) $S(u_i,m;G)$ is an external m-pendant of G for $i = 1, 2$ and for any m with $ec(G)+1 \leq m \leq k$.

(2) $S(u_i,m'+1;G) \subseteq S(u_i,m';G)$ for $i = 1, 2$ and for any m' with $ec(g)+1 \leq m' < k$.

(3) $S(u_1,ec(G)+1;G) \neq S(u_2,ec(G)+1;G)$.

We note that, by Proposition 3.3-(3) of [20-22], we can find a pair of vertices u_1,u_2 satisfying the edge condition for G.

Put

$$G'=G+(u_1,u_2)$$

(We use this notation throughout this section.) Then

$$M_{G'}(u_1,u_2) = M_G(u_1,u_2) + 1 = ec(G) + 1$$

An m-component S of G' that is not an m-component of G is referred to as an *m-augmenting set* of G (with respect to the edge (u_1,u_2)). Any m-component of G' is either an m-component of G or an m-augmenting set of G. Since the addition of the edge (u_1,u_2) to G can increase the number of pairwise edge-disjoint paths by at most one, each m-augmenting set of G is identical to the disjoint union of at least two m-components included in an (m-1)-component of G. Clearly, any m-augmenting set of G is a critical m-component of G'

A pair of distinct vertices $u_1,u_2 \in V(G)$ is said to be *admissible* (with respect to G) if the following (1) and (2) hold:

(1) The pair u_1,u_2 satisfy the edge condition for G.

(2) If $ec(G) = k-1$ and $P^k(G) \geq 4$ then $S(u_1,k;G')$ $(=S(u_2,k;G'))$ is not a k-pendant of G'.

LEMMA 3.2. [20-22]

Suppose that $0 \leq ec(G) < k$. Then we can find an admissible pair u_1,u_2 with respect to G such that

$$EA_k(G) - EA_k(G') = 1.$$

We have proved in [20-22] the following theorem by induction on k-augmentation numbers of graphs.

THEOREM 3.1. [20-22]

For any graph G with $|V(G)| > 1$ and for any fixed $k \geq 2$,

$$R_k(G) = EA_k(G).$$

3.3. THE DATA STRUCTURE

We describe data structures used in an algorithm for finding a minimum solution Z of the problem. We denote

$$n_v = |V(G)|, \quad n_e = |E(G)|$$

for a given graph G.

3.3.1. THE DATA STRUCTURE FOR A GRAPH G

We assume here that G, a multigraph, is given by means of adjacency lists for G^s, a weighted simple graph: it consists of a list head G_{adj} and some *vnodes* representing vertices of G, as in the following declarations, where $N = n_v$ (Figure 1).

```
type
    pnode = ↑ vnode;
    G_adj = array [1..N] of pnode;
    vnode = record
        VNAME, VAL, WT: integer;
        PTR: pnode;
end;
```

Vertices of V(G) are integers 1,...,N and *VNAME* maintains the corresponding integer. *VAL* will be used to maintain current flow values in a maximum flow algorithm. *WT* is set equal to the multiplicity of the corresponding edge in G. The adjacency lists for G^s can be considered as those

for $N(G^s)$, and are used also in a maximum flow algorithm which compute $F(u \rightarrow v) = M_G(u,v)$ for a pair $u,v \in V(G)$ on the network $N(G^s)$, where we put $F(u \rightarrow v) = 0$ if $M_G(u,v) = 0$. For any $Z' \subset Z$, $G + Z'$ will be maintained as adjancy lists for $(G + Z')^s$.

3.3.2. THE DATA STRUCTURE FOR A COMPONENT TREE

We use the following logical data structure, which is referred to as the component tree for G. Let S(m) (S(m)', respectively) denote any m-component of G (of G').

The *component tree* CT(G) is an undirected tree defined by the following (1) and (2):

(1) V(CT(G)) consists of those vertices u_G, $u_{S(m)}$, u_v representing, respectively, V(G), each S(m) $(1 \leqslant m \leqslant k)$, each vertex $v \in V(G)$. Vertices u_G, $u_{S(m)}$, u_v are referred to as the *root*, an *m-component vertex*, a *leaf*, respectively.

(2) For any distinct vertices $u,u' \in V(CT(G))$, there is an edge $(u,u') \in E(CT(G))$ if and only if one of the following (i)-(iii) holds:

(i) u is the root and $u' = u_{S(1)}$.

(ii) $u = u_{S(i)}$ and $u' = u_{S(i+1)}$ such that

$S(i+1) \subseteq S(i)$, $1 \leqslant i \leqslant k$.

(iii) $u = u_{S(k)}$ and $u' = u_v$ such that $v \in S(k)$.

If k = 3 and G is as shown in Figure 2 then CT(G) will be as in Figure 3, where V_1, T_1, S_1 are, respectively, 1-components, 2-components, 3-components of G.

We briefly describe the actual data structure of a component tree. The component tree CT(G) consists of four kinds of data types, *linknode*, *LEVEL*, *CH* and *cnode* as in the following declarations (Figure 4):

type

linknode = ↑ *cnode*;

LEVEL = **array** [0..k] **of** *linknode*;

CH = **array**[1..N] **of** *linknode*;

cnode = **record**

NAME: *integer*;

NEXT, *LLINK*, *RLINK*, *SON*, *TOP*, *F*: *linknode*;

DEG: *integer*;

end;

For each m, $0 \leqslant m \leqslant k$, *LEVEL* [m] is the pointer to the first *cnode* of the list maintaining m-component vertices by means of *NEXT*. This list is called the *m-level*. For each i, $1 \leqslant i \leqslant N$, *CH* [i] is the pointer to the *cnode* corresponding to the leaf i.

NAME has an integer greater than N if the *cnode* represents the root or an m-component vertex, and has one between 1 and N otherwise. Suppose that an m-component S is the disjoint union of (m+1)-components $S_1, \ldots, S_t$, $t \geqslant 1$, $ec(G) \leqslant m \leqslant k$, where we assume that a (k+1)-component means a leaf. Let $R(S)$, $R(S_i)$, denote the *cnodes* representing S, S_i, respectively. Then $R(S_1)$..... $R(S_t)$ are maintained as a doubly linked list by means of *LLINK* and *RLINK*. They are called *sons* of $R(S)$. $R(S)\uparrow.SON$ is pointed to the first son of $R(S)$, and $R(S_i)\uparrow.F$ is pointed to $R(S)$, i = 1 .....t. $R(S)\uparrow.DEG$ is provided for $d_k(S,G)$ if $R(S)$ represents a non-leaf. $R(S)\uparrow.TOP$ is set equal to $R(S)\uparrow.SON$ initially, and is used as the pointer to the first leaf in the sons of $R(S)$ during the procedure is processing $R(S)$ in the construction of CT(G). If $R(S)\uparrow.TOP = nil$ then the partitioning $R(S)$ into $R(S_1), \ldots, R(S_t)$ has been finished. Figure 5 shows a part of the actual data structure corresponding CT(G) of Figure 2.

In the procedure which constructs CT(G), we compute $M_G(u,v)$, $u,v \in V(G)$, by using a maximum flow algorithm, as a subroutine, with a modification such that it will terminate and return the value k whenever the current flow value exceeds k.

Let $\alpha(G)$ denote the time complexity of a maximum flow algorithm with such a modification to compute $F(u \rightarrow v;G) = M_G(u,v)$. Then CT(G) can be constructed in $O(n_v^2 \alpha(G))$ time. For example it is $O(n_v^2 \cdot Tn_v n_e)$ if we use the Dinic's algorithm and is $O(n_v^2 \cdot Tn_v^2)$ if we use the algorithm pro-

posed by Malhotra, Kumar and Maheshwari (MKM, for short), where $T = \min\{k, n_v\}$. (For the detail, see [4, 13].)

(In [9] it is shown that we can determine $F(u \rightarrow v; G)(= M_G(u,v))$ for all pairs $u,v \in V(G)$ by using a maximum flow algorithm only $O(n_v)$ times, instead of $O(n_v^2)$ times as described above. In this paper, however, we do not take advantage of this result simply for ease of implementation. The time complexity for constructing CT(G) would be $O(n_v \cdot n_v^2 ne)$ in Dinic's case and $O(n_v \cdot n_v^3)$ in MKM's case.)

3.3.3. COMPUTATION ON CT(G)

Suppose that we have CT(G). We describe the time complexity of the following computation (1) - (5) on CT(G)

(1) $d_k(S(m),G)$, $D_k(G)$, ec(G) and $P^{(m)}(G)$.

The computation of the following (i) - (iv) can be done in $O(k(n_v + n_e))$ time:

(i) The degree $d_k(S(m),G)$ for every m-component S(m) of G, m = 1,...,k.

(ii) $D_k(G)$ (the demand of G).

(iii) ec(G) (the edge-connectivity of G).

(iv) $p^{(m)}(G)$ (the number of m-pendants of G) with m = ec(G)+1.

(2) Finding a pair u_1, u_2 satisfying the edge condition.

Let S be an external m-pendant of G. Proposition 3.3 of [20-22] shows that S includes at least one external (m+1)-pendant of G. First, suppose that there are distinct (m+1)-components $S_1, S_2 \subset S$ of G. Since

$$M_G(v_1, v_2) = m \quad \text{for } \forall v_i \in S_i, \ i = 1, 2,$$

we have

$$d_k(S_i, G) = |K(S_i, G)| \geq m, \ i = 1, 2.$$

If

$$d_k(S_i, G) = m \text{ for either } i = 1 \text{ or } i = 2$$

then $K(S_i, G)$ is a (S_1, S_2)-cut of G, meaning that S_i is an external (m+1)-pendant of G. If

$$d_k(S_i, G) > m$$

then S_i is not an (m+1)-pendant of G. Suppose that S is also an (m+1)-component of G. Then, clearly, S is an external (m+1)-pendant of G.

Therefore if we specify a *cnode* of representing an external m-pendant S(m) of G, we can find in $O(kn_v)$ time the *cnode* representing a leaf u such that

$$u \in S(k) \subseteq \cdots \subseteq S(m+1) \subseteq S(m),$$

where each S(t), $m+1 \leq t \leq k$, is an external t-pendant of G. Hence we can find a pair u_1, u_2 satisfying the edge condition for G in $O(kn_v)$ time.

(3) Constructing adjacency lists for $(G')^S$.

We can construct adjacency lists for $(G')^S$ in $O(n_v)$ time, where $G' = G + (u_1, u_2)$.

(4) Finding an admissible pair.

Suppose that we have a pair u_1, u_2 satisfying the edge condition for G and adjacency lists for $(G')^S$, where $G' = G + (u_1, u_2)$. It suffices to consider the case where $k = ec(G) + 1$ and $P^k(G) \geq 4$.

We choose a vertex v_j from each k-component $S_j (\neq S(u_i, k;G); i=1,2)$ of G, and compute $M_{G'}(u_i, v_j)$. If $M_{G'}(u_i, v_j) \geq k$ then *DEG* of every son of the *cnode* representing $S(v_j, k;G)$ is set to 1, which indicates $S(v_j, k;G) \subseteq S(u_i, k;G')$. Then we compute $d_k(S(u_1, k;G'), G')$ in $O(|E(G^S)|)$ time by counting the total weight of edges $(u,v) \in E(G') \cap K(S(u_1, k;G'), G')$.

If $d_k(S(u_1, k;G'), G') < k$ then $S(u_1, k;G')$ is a k-pendant of G', and we choose another vertex u_2' from a k-pendant not included in $S(u_1, k;G')$. This choice is done in $O(n_v)$ time. Lemma 3.2 of [20-22] shows that the pair u_1, u_2' is an admissible pair. Thus we can find an admissible pair with respect to G in $O(n_v \alpha(G') + |E(G')|)$ time if we compute $M_{G'}(u_1, v_j)$ by means of a maximum flow

algorithm.

(5) An algorithm proposed in [20-22].

The proof of Theorem 3.1 shows an algorithm for finding a minimum solution of the problem. First construct the initial data structure and compute the initial data (i)-(iv):

(i) $d_k(S(m),G)$ for every $S(m)$, $m=1,\ldots,k$.

(ii) $D_k(G)$.

(iii) $ec(G)$.

(iv) $P^m(G)$ with $m = ec(G) + 1$.

Then repeat the following (a), (b) by $EA_k(G)$ times.

(a) Finding an admissible pair u_1,u_2 with respect to G and the construction of $CT(G')$, $G' = G + (u_1,u_2)$.

(b) The computation of the following (i)-(iii) for G'.

(i) $d_k(S(m)',G')$ for every m-component $S(m)'$ of G', $m=1,\ldots,k$.

(ii) $ec(G')$.

(iii) $P^m(G')$ with $m=ec(G')+1$

Suppose that we use Dinic's maximum flow algorithm. Then the initial data structure and the initial data (i)-(iv) can be obtained in $O(kn_v^3n_e)$ time. (a), (b) for G' can be done in $O(kn_v^3|E(G')|)$ time. Since we have

$$|Z| = EA_k(G) \leq kn_v$$

and

$$|E(G)| + |E(G')| + \ldots + |E(G+Z)| = n_e + (n_e+1) + \ldots + (n_e+EA_k(G)) \leq (kn_v+1)(n_e+kn_v)$$

for any solution Z, the total time is

$$O(k^2 n_v^4(kn_v + n_e).$$

If we use MKM's maximum flow algorithm then the total time is

$$O(kn_v(n_e+kn_v^4)).$$

The most time-consuming part of this algorithm is constructing a component tree, which is repeated each time a new edge is added. In the next section we will propose an improved algorithm in which constructing a component tree is repeated at most k-1 times instead of $EA_k(G)(\leqslant kn_v)$ times mentioned above.

4. AN IMPROVEMENT OF THE ALGORITHM

We consider an improvement of the algorithm mentioned in 3.3.3-(5). The idea of the improvement is as follows.

The previous algorithm reconstructs the component tree each time we find only one edge to be added. We can expect a more efficient algorithm if there is an easy way to find as many edges to be added as possible before reconstructing component trees: it may reduce both time spent to find such edges and the number of times of reconstructing component trees. We will describe more precisely.

Suppose that

$$ec(G)<k,\ k \geqslant 2,$$

and let Z be a solution obtained by the algorithm mentioned in 3.3.3-(5). Since each edge of Z joins a pair of vertices satisfying the edge condition, Z has a partition

$$Z = Z(ec(G) + 1) \cup \cdots \cup Z(k),$$

$$Z(i) \cap Z(j) = \phi(i \neq j),\ Z(i) \neq \phi\ (ec(G) + 1 \leqslant i \leqslant k)$$

such that

$$ec(G_i) = ec(G_{i-1}) + 1 = i.$$

$$ec(G_i - e) = ec(G_{i-1}) \text{ for } ve \in Z(i).$$

where

$$G_{ec(G)} = G,\ G_i = G_{i-1} + Z(i),\ i = ec(G)+1, \ldots, k.$$

We will consider two procedures, *connect* in 4.1 and *find* in 4.2: the first one determines Z(1) for G with ec(G) = 0, and the second one does Z(m+1), m=ec(G), for G with ec(G) > 0. The procedure *find* is used repeatedly to determine Z(m+).....Z(k) in this order such that reconstructing the component tree will be done only between Z(j) and Z(j+1) for each j, $ec(G)+1 \leqslant j \leqslant k-1$. In 4.3 we describe the outline of an improved algorithm and estimate its time complexity.

REMARK 4.1.

(1) If ec(G) = 0 then $|Z(1)| \geqslant P^1(G) - 1$ (Note that $P^1(G)$ is equal to the number of components of G.)

(2) If ec(G) > 0 then $|Z(i)| \geqslant [P^1(G_{i-1})/2]$ for each i, $ec(G)+1 \leqslant i \leqslant k$.

Let m denote any fixed integer such that

$$ec(G) \leqslant m \leqslant k-1,$$

and put

$$H = G_m/\pi(m+1),$$

where $\pi(m+1)$ denotes the partition of $V(G_m)$ into (m+1)-components of G_m. Let ζ_{m+1} denote the mapping of G_m induced by $\pi(m+1)$. We have

$$d(u,H) \geqslant m,\ M_H(u,v) = m$$

for any $u, v \in V(H)$. A vertex u with d(u,H) = m is called an (m+1)- *pendant* or a *pendant* of H, and $P^{m+1}(H) \geqslant 2$.

REMARK 4.2.

If u is a pendant of H then $K(\{u\},H)$ is a (u,v)-cut of H for any $v \in V(H)$ and $\zeta_{m+1}^{-1}(u) \in \pi(m+1)$ is an external (m+1)-pendant of G_m. Any (m+1)-pendant S of G_m is external and $\zeta_{m+1}(S) \in V(H)$ is a pendant of H.

Let

$$Y = \{y_1, \ldots, y_q\} \quad (q = p^{m+1}(H))$$

denote the set of all pendants of H, and put

$$r = [q/2].$$

Put

$$V(e) = \{u,v\} \text{ for an edge } e=(u,v),$$

and

$$V(E) = \bigcup_{e \in E} V(e) \text{ for a set } E \text{ of edges.}$$

Let E be a set of edges. We call E an *attachment* (for H) if and only if the following (1)-(4) hold:

(1) $V(E) \subset Y$.

(2) $E \cap E(H) = \phi$.

(3) $V(e) \neq V(e')$ for $e,e' \in E$, $e \neq e'$.

(4) There is at most one pair $f, f' \in E$ such that $|V(f) \cap V(f')| = 1$.

4.1. THE PROCEDURE *CONNECT* (ec(G)=0)

We consider the procedure *connect*, which determines $\mathcal{A}(H)$ and constructs adjacency lists for $(G^*+\mathcal{A}(H))$ with $ec(G)=0$. Let

$$m = ec(G) (=0).$$

and consider

$$H = G/\pi(1)$$

Then

$$V(H) = Y_q \quad q \geq 2, \quad E(H) = \phi.$$

Define a set of edges

$$Z_H = \{e_i = (y_i, y_{i+1}) : i = 1, \ldots, q-1\},$$

where

$$Z_H \cap E(H) = \phi.$$

Clearly

$$ec(H + Z_H) = 1, \quad ec(H + Z_H') = 0 \text{ for } \forall Z_H' \subset Z_H.$$

Now we will show how to choose a pair of vertices satisfying the edge condition. For each i, $1 \leq i \leq q$, put

$$S_i = \zeta_1^{-1}(y_i) \in \pi(1)$$

There are two cases concerning each S_i.

(1) S_i is a k-component of G.

(2) S_i is a critical t-component of G where $1 \leq t < k$.

Proposition 3.3 of [20-22] shows that, in both (1) and (2), there is a sequence

$$S_i(k) \subseteq \quad \subseteq S_i(t+1) \subseteq S_i$$

for each $i = 1, 2, \ldots$ where $S_i(t')$ denotes an external t'-pendant of G, $t' = t+1, \ldots, k$.

$S_i \cap S_i = S_i$ if S_i is a k-component.

$S_i(t+1) \cap S_i + 1 = \phi$ if S_i is a critical t-component, $t < k$

Choose vertices u_{xj}, x=1,....,q-1; j=1,2, as follows:

(i) If S_x (S_{x+1}, respectively) satisfies (1) then
$u_{x1} \in S_x$ ($u_{x2} \in S_{x+1}$).
(ii) If S_x (S_{x+1}) satisfies (2) then
$u_{x1} \in S_{x1}(k)$ ($u_{x2} \in S_{x+1,2}(k)$).

Put

$$e_x = (u_{x1}, u_{x2}),\ x=1,....,q-1,$$

where

$$e_x \notin E(G).$$

We consider the case where x = 1:

$$G,\ G' = G + e_1.$$

Clearly, the pair u_{11}, u_{12} satisfies the edge condition for G. We will show that if $q \geq 3$ then the pair u_{21}, u_{22} satisfies the edge condition for G'. If this is shown then the discussion for x=1 can be applied to the general case:

$$G + \{e_1, \ldots, e_{x-1}\},\ G + \{e_1, \ldots, e_{x-1}, e_x\},\ x \geq 3.$$

Proposition 3.6 and Lemma 3.4 show the following (a)-(c):

(a) $S(u_{11}, 1;G') = S(u_{12},1;G')(=S_1 \cup S_2)$, and it is the only 1-augmenting set of G with respect to e_1.

(b) Any m'-component S' of G' is also an m'-component of G if $m' > 1$ or if $S' \neq S(u_{11},1;G')$ with $m' = 1$.

(c) For any m'-component S of G', $ec(G) \leq m' \leq k$.

$$D_k(S,G)-D_k(S,G') = \begin{cases} 2 & \text{if } V(e_1)\subset S \text{ and either } P^{m'}(G)\neq 3 \text{ or } m' \neq k. \\ 1 & \text{if } |V(e_1)\cap S| = 1, \text{ or if } V(e_1)\subset S \text{ and } P^{m'}(G)=3 \text{ with } m'=k. \\ 0 & \text{otherwise.} \end{cases}$$

Hence we have (d), (e):

(d) If $S_2 = S_{21}(k)$ then S_2 is also a k-component of G' with $D_k(S_2, G') = D_k(S_2, G)-1$ $(=k-1)$

(e) If S_2 is a critical t-component of G, $1 \leqslant t < k$, then (i), (ii) hold.

(i) $S_{21}(t')$ is an external t'-pendant of G' with $D_k(S_{21}(t'),G') = D_k(S_{21}(t'),G)$ for each t', $t+1 \leqslant t' \leqslant k$.

(ii) If $t \geqslant 2$ then S_2 is an external t''-pendant of G' for each t'', $2 \leqslant t'' \leqslant t$.

It follows that there is a sequence of external pendants of G'

$$S_{21}(k) \subseteq \cdots \subseteq S_{21}(t+1) \subset S_1 \bigcup S_2$$

such that if S_2 is a critical t-component of G with $2 \leqslant t < k$ then S_2 is an external t''-pendant of G' and

$$S_{21}(t+1) \subseteq S_2 \subset S_1 \bigcup S_2$$

for each t'', $2 \leqslant t'' \leqslant t$. Thus the pair u_{21}, u_{22} satisfies the edge condition for G', and we obtain the following proposition.

PROPOSITION 4.1.

Let

$$E_1 = \{e_1, \ldots, e_{q-1}\},\ e_i = (u_{i1}, u_{i2}),\ 1 \leqslant i \leqslant q-1.$$

Then we can set

$$Z(1) = E_1.$$

The procedure *connect* repeats two procedures: finding a pair u_{i1}, u_{i2} satisfying the edge condition for the current graph G in $O(kn_v)$ time and then constructing adjacency lists for $(G^S + (u_{i1}, u_{i2}))^S$ in $O(n_v)$ time. Thus the procedure *connect* finds $Z(1)$ and constructs adjacency lists for $(G^S + Z(1))^S$ in $O(kn_v^2)$ time.

$CT(G + E_1)$ is easily obtained from $CT(G)$: coalescing all 1-component vertices of $CT(G)$ into one 1-component vertex, merging corresponding sons lists into one list, and changing degrees of

corresponding component vertices in $O(kn_v)$ time (by means of (c)).

4.2. THE PROCEDURE *FIND* (ec(G) > 0)

We consider the procedure *find*, which determines Z(m+1) and constructs adjacency lists for $(G^S + Z(m+1))^S$ with $m = ec(G) > 0$. In 4.2.1 and 4.2.2, we consider the procedure *hfind*, which determines a minimum attachment Z_H for $H = G/\pi(m+1)$ such that $ec(H+Z_H) = m+1$. In 4.2.3, we consider how to determine an edge $(u',v') \in Z(m+1)$ from each edge $(u,v) \in Z_H$. In 4.2.4 we describe the procedure *find*, a modified version of the procedure *hfind*.

Let E be any attachment for H. For each edge $e = (u,v) \in E$, H+E has a new (m+1)-component, denoted by A(e,H+E), containing V(e), since

$$M_{H+e}(u,v) = M_H(u,v) + 1 = m+1 \leqslant M_{H+E}(u,v).$$

A(e,H+E) is referred to as the (m+1)-*augmenting set for e* (with respect to E).

First we show the following proposition for an attachment E for H.

PROPOSITION 4.2.

Suppose that an attachment F for H satisfies the following (1), (2):

$$\begin{cases} (1)\ V(E) = Y \\ (2)\ H + E \text{ has an } (m+1)\text{-component } A \text{ such that } V(E) \subseteq A. \end{cases}$$

Then

$$A = V(H).$$

PROOF.

Assume that

$$A \subset V(H).$$

Corollary 3.1 of [20-22] shows that H + E has an (m+1)-component S such that

$$S \subseteq V(H) - A.$$

Proposition 3.1 of [20-22] shows that H + E has an (A,S)-cut K with |K| = m. Proposition 3.3 of [20-22] shows that B(S,K;H+E) contains an (m+1)-pendant S' of H + E. We can show, by using Theorem 3.1 of [20-22], S' is also an (m+1)-pendant of H. Hence S' = {v} for some vertex $v \in V(H)-A$. It follows that

$$(V(H)-A) \bigcap Y \neq \phi,$$

a contradition.

Q.E.D.

We will show, in 4.2.1 and 4.2.2, that we can find an attachment E,

$$E = \{e_1, \ldots, e_r\} \quad (r = \lfloor q/2 \rfloor),$$

satisfying the following (i) - (iii):

(i) For each i, $1 \leq i \leq r-1$,

$A(e_i,H_i) \bigcap A(e_{i+1},H_{i+1}) \neq \phi$ if $r \geq 2$

where

$H_0 = H$, $H_j = H_{j-1} + e_j$, $j = 1, \ldots, r$

(ii) $V(E) = Y$.

(iii) $V(e_i) \bigcap V(e_j) \neq \phi$ if and only if q is odd, $i = r-1$ and $j = r$.

If such E exists then $A(e_r,H_r)$ is an (m+1)-component of H + E and

$$V(H) \subseteq A(e_r,H_r).$$

Proposition 4.2 shows that

$$A(e_r,H_r) = V(H).$$

4.2.1. THE CASE WHERE q = 2

If $q = 2$ then let

$$Z_H = \{e_1 = (y_1, y_2)\},\ e_1 \in E(H).$$

Clearly Z_H is an attachment for H. It is easy to see that Z_H and $Z(e_1,H_1)$ satisfy Proposition 4.2-(1), (2), showing that

$$A(e_1,H_1) = V(H).$$

Thus we obtain the following.

PROPOSITION 4.3.

If $q = 2$ then

$$A(e_1,H_1) = V(H)$$

for $e_1 = (y_1,y_2) \in E(H)$.

4.2.2. THE CASE WHERE q ⩾ 3

Let E be any attachment for H such that there are vertices

$$v_{ij} \in Y - V(E),\ i,j = 1, 2,$$

where v_{11},v_{12},v_{21} are pairwise distinct, and if v_{22} is equal to one of the rest then we assume, without loss of generality, that

$$v_{22} = v_{12}.$$

Put

$$L = H + E,\ e = (v_{11},v_{12}),\ e' = (v_{21},v_{22}),$$

where

$$e, e' \in E(L).$$

Put

$$A(e) = A(e,L + \{e,e'\}),\ A(e') = A(e',L + \{e,e'\}),$$

$$Ae = A(e,L + e'),\ Ae' = A(e',L + e').$$

Clearly,

$$Ae \subset A(e),\ Ae' \subset A(e'),$$

and

$$Ae \bigcup Ae' \subset A(e) = A(e') \text{ if } A(e) \bigcap A(e') \neq \phi.$$

In the following we first consider the case I where

$$A(e) \bigcap A(e') = \phi,$$

i.e., $L + \{e,e'\}$ has an $(A(e),A(e'))$-cut consisting of m edges. Then we proceed to another case II where

$$A(e) \bigcap A(e') \neq \phi.$$

CASE I. $A(e) \bigcap A(e') = \phi$. (Then $V(e) \bigcap V(e') = \phi$.)

Let K be any fixed $(A(e),A(e'))$-cut of $L + \{e,e'\}$, and let B_i, $i = 1, 2$, denote the K-block of $L + \{e,e'\}$ such that

$$V(e) \subset A(e) \subset V(B_1),\quad V(e') \subset A(e') \subset V(B_2).$$

(In the following discussion, we write B_i instead of $V(B_i)$ for simplicity.) We note that K is also an $(A(e), A(e'))$-cut of L, of $L + e$ or of $L + e'$.

Let f, f' be two edges defined by either

$$f = (v_{11},v_{21}),\ f' = (v_{12},v_{22})$$

or

$$f = (v_{11},v_{22}),\ f' = (v_{12},v_{21}),$$

where

$$\{f,f'\} \bigcap E(L) = \phi.$$

Let b be any fixed vertex from $\{v_{21},v_{22}\}$ and put

$$I = (L + e) < b, B_2 >.$$

We note that

$$V(I) = B_1 \bigcup \{b\},\ I = (L + \{e,e'\}) < b, B_2 >.\ \text{(See Fig. 6.)}$$

PROPOSITION 4.4.

Suppose that $A(e) \bigcap A(e') = \phi$. Then

$$M_{L'}(u,u') = M_L(u,u')$$

for any $u,u' \in V(L)$, where $L' = L + e$ or $L' = L + \{e,e'\}$.

PROOF.

Since

$$M_{L+e}(v_{11},v_{21}) = M_{L+\{e,e'\}}(v_{11},v_{21}) = m = |K|.$$

Theorem 3.1 of [20-22] shows that

$$M_{L'}(u,u') = M_L(u,u').$$

Q.E.D.

PROPOSITION 4.5.

Suppose that $A(e) \bigcap A(e') = \phi$. Then

$$M_{L'}(v,v') \leq M_{L+\{f,f'\}}(v,v')$$

for any $v,v' \in B_1$, where $L' = L + e$ or $L' = L + \{e,e'\}$.

PROOF.

It suffices to say that

$$M_{L+e}(v,v') \leq M_{L+\{f,f'\}}(v,v').$$

Since L-K has just two components whose sets of vertices are B_1 and B_2, L-K has a (v_{j1},v_{j2})-path Q_j such that

$$E(Q_j) \bigcap K = \phi, \; j = 1, 2.$$

Hence $L + \{f,f'\}$ has a circuit C such that

$$E(C) = E(Q_1) \bigcup E(Q_2) \bigcup \{f,f'\}.$$

Let P_1 denote the (v_{11},v_{12})-subpath of C defined by

$$E(P_0) = E(C) - E(Q_1).$$

Let $P_1, \ldots, P_t$ denote any fixed set of (v,v')-path of $L + e$, where $t = M_{L+e}(v,v')$. If none of them passes through e then all of them are paths of $L + \{f,f'\}$ and the proposition follows. Suppose that $e \in E(P_1)$. Let P_{1j}, $j = 1, 2$, denote the two subpaths of $P_1 - e$, where we may have

$$E(P_{11}) = \phi \text{ or } E(P_{12}) = \phi.$$

P_{11}, P_{12} and all other $P_i (i \geq 2)$ are paths of $L + \{f,f'\}$. Let P_1' denote the (v,v')-path of $L + \{f,f'\}$ defined by joining P_{11}, P_0, P_{12}. Then

$$E(P_1') \bigcap E(P_i) = \phi \quad \text{if} \quad i \geq 2.$$

That is,

$$M_{L+e}(v,v') \leq M_{L+\{f,f'\}}(v,v').$$

Q.E.D.

COROLLARY 4.1.

Suppose that $A(e) \bigcap A(e') = \phi$. Then $L + \{f,f'\}$ has the $(m+1)$-component A such that

$$Ae \subseteq A(e) \subseteq A.$$

Put

$$f_1 = (v_{11}, v_{21}), \; f_3 = (v_{11}, v_{22}), \; f_2 = (v_{12}, v_{22}), \; f_4 = (v_{12}, v_{21}),$$

$$A(f_i) = \begin{cases} A(f_i, L + \{f_1, f_2\}) & \text{if } 1 \leq i \leq 2. \\ A(f_i, L + \{f_3, f_4\}) & \text{if } 3 \leq i \leq 4. \end{cases}$$

PROPOSITION 4.6.

Suppose that we have

$$A(e) \bigcap A(e') = \phi \quad \text{and} \quad A(f_1) \bigcap A(f_2) = \phi.$$

Then

$$A(f_3) \bigcap A(f_4) \neq \phi, \text{ i.e., } A(f_3) = A(f_4).$$

PROOF.

$L+\{f_1,f_2\}$ has an $(A(f_1),A(f_2))$-cut consisting of m edges. Let K' denote any fixed $(A(f_1),A(f_2))$-cut of $L + \{f_1,f_2\}$. Then $K' \neq K$. K' is also an $(A(f_1),A(f_2))$-cut of L, of $L + f_1$ or of $L + f_2$. Let B_i' denote the K'-block of $L + \{f_1,f_2\}$ containing $V(f_i)$, $i = 1, 2$. We have

$$v_{ij} \in B_i \bigcap B_j', \quad i,j = 1, 2.$$

Suppose that

$$V(K) \bigcap B_1 = \{v\}.$$

Then any (v_{1j},v_{2j})-path, j=1,2, of L and of $L + \{e,e'\}$ passes through v. Each of m edge-disjoint (v_{1j},v_{2j})-paths is decomposed into two subpaths: the (v_{1j},v)-subpath and the (v,v_{2j})-subpath, showing that

$$M_{L+\{f_1,f_2\}}(v_{1j},v) = m + 1, \quad i,j = 1,2.$$

That is,

$$v \in A(f_1) \bigcap A(f_2),$$

a contradiction. Similarly we can show that

$$|V(K) \bigcap B_i| \geq 2, \quad i = 1, 2, \quad |V(K') \bigcap B_j'| \geq 2, \quad j = 1, 2.$$

Let

$$K = \{e_i: \ 1 \leq i \leq m\}, \quad K' = \{e_i': \ 1 \leq i \leq m\},$$

and let $P_{31}, \ldots, P_{3m}$ denote any set of m edge-disjoint (v_{11},v_{22})-paths of L. Then

$$|E(P_{3i}) \bigcap K| = |E(P_{3i}) \bigcap K'| = 1, \quad i = 1,\ldots,m.$$

If we assume that there is $e \in K \bigcup K'$ such that

$$V(e) \bigcap (B_2 \bigcap B_1') \neq \phi, \quad V(e) \bigcap (B_1 \bigcap B_2') \neq \phi$$

then some P_{3i} passes through e, showing a contradiction that

$$|E(P_{3i}) \bigcap K| \geq 2 \text{ or } |E(P_{3i}) \bigcap K'| \geq 2.$$

Hence no such $e \in K \bigcup K'$ exists. Let $P_{41}, \ldots, P_{4m}$ denote any set of m edge-disjoint (v_{21},v_{12})-paths of L. Then we can similarly show that there is no $e \in K \bigcup K'$ such that

$$V(e) \bigcap (B_i \bigcap B_i') \neq \phi, \quad i = 1, 2.$$

For each i, i = 1, 2, put

$$K_i = \{e_j \in K: \ V(e_j) \subset B_i'\}, \ K_i' = \{e_j' \in K': \ V(e_j') \subset B_i\}.$$

Clearly,

$$K = K_1 \bigcup K_2, \ K' = K_1' \bigcup K_2', \ K_1 \bigcap K_2 = K_2' \bigcap K_2' = \phi.$$

$$K_i \neq \phi, \ K_i' \neq \phi, \ i = 1,2, \ K \bigcap K' = \phi.$$

We also have

$$|K_1| = |K_2| = |K_1'| = |K_2'|,$$

showing that m is even. Put

$$x = m/2.$$

For each P_{3i}, i = 1.....m, put

$$P_{3i}^{(t)} = P_{3i}[B_t \bigcap B_t'], \ t = 1, 2.$$

Similarly, for each P_{4i}, i = 1.....m, put

$$P_{4i}^{(1)} = P_{4i}[B_2 \bigcap B_1'], \ P_{4i}^{(2)} = P_{4i}[B_1 \bigcap B_2'].$$

We note that these 4m subpaths are pairwise edge-disjoint. It follows that L has pairwise edge-disjoint (v_{11},v_{21})-path Q_i, i = 1.....m, defined by these 4m subpaths and $K \bigcup K'$. (See Figure 7).

Let Q_{m1}(Q_{m2}, respectively) denote the (v_{11},v_{12})-subpath (the (v_{22},v_{21})-subpath) of Q_m, and let R_{m1} (R_{m2}, respectively) denote the (v_{11},v_{21})-path of L + $\{f_3,f_4\}$ defined by joining

$$Q_{m1}, f_4 \ (Q_{m2}, f_3).$$

The paths $Q_1, \ldots, Q_{m-1}, Q_{m1}, Q_{m2}$ are pairwise edge-disjoint (v_{11},v_{21})-paths of L + $\{f_3,f_4\}$, showing that

$$A(f_3) \bigcap A(f_4) \neq \phi.$$

Q.E.D.

The next corollary follows from Corollary 4.1 and Proposition 4.6.

COROLLARY 4.2.

Suppose that $A(e) \bigcap A(e') = \phi$. Then

$$\begin{cases} Ae \subseteq A(e) \subset A(f_1) = A(f_2) & \text{if } A(f_1) \bigcap A(f_2) \neq \phi. \\ Ae \subseteq A(e) \subset A(f_3) = A(f_4) & \text{otherwise.} \end{cases}$$

CASE II. $A(e) \bigcap A(e') \neq \phi$.

Put

$$e = (v,w),\ e' = (v',w'),\ L' = L + e.$$

Suppose that there are distinct vertices

$$v'',w'' \in Y - (V(E) \bigcup V(e) \bigcup V(e'))$$

such that

$$A(e',L'+\{e',e''\}) \bigcap A(e'', L' + \{e',e''\}) = \phi,$$

where

$$e'' = (v'',w'') \in E(L).$$

Corollary 4.2 shows that we can find a pair of edges f,f' such that

$$A(f,L' + \{f,f'\}) \bigcap A(f',L' + \{f,f'\}) \neq \phi,$$

where

$$V(f) \bigcup V(f') = V(e') \bigcup V(e''),$$

$$V(f) \bigcap V(f') = \phi.$$

We assume, without loss of generality, that

$$f = (v',w''),\ f' = (v'',w').$$

PROPOSITION 4.7.

If

$$A(e,L') \bigcap A(f,L' + f) = \phi$$

then

$$A(e,L') \bigcap A(f',L' + f') \neq \phi.$$

PROOF.

First we note that

$$A(e,L') \subseteq A(e,L' + e') = A(e',L' + e') \subset A(f,L' + \{f,f'\}) = A(f',L' + \{f,f'\}) \text{ (by Corollary 4.2).}$$

$$A(e,L') \subseteq A(e,L' + f), \quad A(e,L'+f) \bigcap A(f,L'+f) = \phi.$$

$L' + f$ has an $(A(e,L' + f), A(f,L' + f))$-cut K with $|K| = m$. Let B_e, B_f denote the K-blocks of $L' + f$ such that

$$A(e,L') \subset Be, \quad A(f,L' + f) \subset B_f.$$

We can assume, without loss of generality, that

$$w' \in B_e, \quad v'' \in B_f. \text{ (See Fig. 8).}$$

Put

$$K_{e'} = K \bigcup \{e'\}, \quad K_{f'} = K \bigcup \{f'\}.$$

Since

$$M_{L'+e'}(v',w') = M_{L'+f'}(v'',w') = m + 1.$$

$K_{e'}$ or $K_{f'}$ is a (v',w')-cut of $L' + e'$ or a (v'',w')-cut of $L'+f'$, respectively. B_e, B_f are the $K_{e'}$-blocks of $L' + e'$ and the $K_{f'}$-blocks of $L' + f'$. Let v_f be any fixed vertex of B_f. Then $(L' + e')\langle v_f, B_f \rangle$ is isomorphic to $(L' + f')\langle v_f, B_f \rangle$. Put

$$L'' = (L' + e')\langle v_f, B_f \rangle.$$

Then, by Theorem 3.1 of [20-22],

$$M_{L'+e'}(u,u') = M_{L''}(u,u') = M_{L'+f'}(u,u')$$

for any $u,u' \in V(L'') = Be \bigcup \{v_f\}$. This shows that

$$M_{L'+e'}(w,x') \geq m + 1,$$

since

$$M_{L'+e}(w,w') \geq m+1$$

That is,

$$A(e,L') \bigcap A(f',L'+f') \neq \phi$$

Q.E.D.

COROLLARY 4.3.

$$A(e,L'+\{f,f'\}) = A(f,L'+\{f,f'\}) = A(f',L'+\{f,f'\})$$

We will show, by using Propositions 4.3, 4.6, and 4.7, that if $ec(G) > 0$ then we can find a sequence of edges

$$e_1, \ldots, e_r, \quad r = |q,2|$$

such that

$$A(e,H) \subset A(e_1,H_1), \quad i = 1, \ldots, r-1$$

where

$$H_1 = H, \quad H_{i+1} = H + e_{i+1}$$

We first describe the procedure *hfind* where we put for each i, $1 \leq i \leq r-1$

$$A(f,\ldots) = A(f,H_i) + \{f,f'\}$$

$$A(f',\ldots) = A(f',H_i) + \{f,f'\}$$

for two edges f, f' such that

$$V(f) \cup V(f') \subseteq V(\ldots), \quad V(f) \neq V(f')$$

```
procedure find
01      begin
02          H = H_1 = 1 ; E_H = ∅ ; x = x_1 ; x' = x_2 ; t = (x, w);
03          while i ≤ r − 1 do begin
04              i = x_{i+1}
05              if (q is odd and i = r − 1) then x' = x_{i−1} else w = y_{2i+2};
06              t = (i, x');
07              if A(t) ⋂ A(H_1) = ∅ then begin
08                  t = (i, x'); H = (x, x');
09                  if A(t) ⋂ A(H_1) = ∅ then begin
                        t = (x, x'); H = (i, x');
                    end
                end
                if 2 ≤ i ≤ r − 1 then
                    if x ∈ H_{i−1} ⋂ A(H_{i+1}) = ∅ then begin
                        H = H_1 = 1; t = (i, 1); H = 1;
                    end
                H = H_1 = H_{i+1} + e; E_H = E_H ⋃ e;
                if i = r − 1 then begin
                    e = (i, H_{i−1} = H + e; E_H = E_H ⋃ e;
                end
                i = i + 1;
            end
        end
```

The part from line 4 through line 22 is called the *i-phase* for each i, $1 \leqslant i \leqslant r-1$.

Let f_i', f_{i+1} denote the two edges (v,w), (v',w'), respectively, that we have at the beginning of the i-phase, and let e_i, f_{i+1}' denote those edges obtained at the end of the i-phase, where

$$f_1' = (y_1, y_2), \ f_r' = e_r.$$

$$V(f_i') \bigcup V(f_{i+1}) = V(e_i) \bigcup V(f_{i+1}').$$

For each j, $1 \leqslant j \leqslant r$, put

$$Y_j = \begin{cases} Y & \text{if } q \text{ is odd and } j = r, \\ \{y_1, y_2, \ldots, y_{2j-1}, y_{2j}\} & \text{otherwise.} \end{cases}$$

PROPOSITION 4.8.

$$A(e_i,H_i) \subset A(e_{i+1},H_{i+1}) \text{ for each } i, \ 1 \leqslant i \leqslant r-1.$$

PROOF.

Proposition 4.6 shows that, at the end of the 1-phase, we have two edges

$$e_1, \ f_2'$$

such that

$$A(e_1,H_1) \subset A(e_1,H_1 + f_2') = A(f_2',H_1 + f_2').$$

Suppose that, at the end of the (i-1)-phase, $2 \leqslant i \leqslant r-1$, we have two edges

$$e_{i-1}, \ f_i'$$

such that

$$A(e_{i-1},H_{i-1}) \subset A(e_{i-1},H_{i-1} + f_i') = A(f_i',H_{i-1} + f_i').$$

We consider the i-phase. At the beginning of the phase we have two edges

$$f_i', \ f_{i+1}.$$

Put

$$L' = H_{i-1}, \ L'' = H_{i-1} + f_i',$$

and suppose that

$$A(f_i', L'' + f_{i+1}) \bigcap A(f_{i+1}, L'' + f_{i+1}) = \phi.$$

Then Proposition 4.6 assures that, at line 12, we obtain two edges f, f' such that

$$V(f) \bigcup V(f') = V(f_i') \bigcup V(f_{i+1}),$$

$$V(f) \neq V(f')$$

$$A(f, L' + \{f, f'\}) = A(f', L' + \{f, f'\}).$$

Proposition 4.7 also assures that we have

$$A(e_{i-1}, L') \bigcap A(f, L' + f) \neq \phi \text{ or } A(e_{i-1}, L') \bigcap A(f', L' + f') \neq \phi.$$

Hence, at line 20, we obtain two edges

$$e_i, f_{i+1}'$$

such that

$$A(e_{i-1}, L') \subset A(e_i, L' + e_i), \; L' + e_i = H_i$$

At the end of the (r-1)-phase, we obtain two edges

$$e_{r-1}, f_r' = e_r$$

such that

$$A(e_{r-1}, H_{r-1}) \subset A(e_{r-1}, H_{r-1} + f_r') = A(f_r', H_{r-1} + f_r') = A(e_r, H_r).$$

Q.E.D.

REMARK 4.3.

(1) For each i, $1 \leq i \leq r-1$,

$$\begin{cases} |V(e_i) \bigcap V(e_{i+1})| = 1 & \text{if } q = \text{odd and } i = r-1, \\ V(e_i) \bigcap V(e_{i+1}) = \phi & \text{otherwise.} \end{cases}$$

$$A(e_i, H_i + f_{i+1}') \bigcap Y = Y_{i+1},$$
$$A(e_i, H_i) \bigcap Y = V(e_1) \bigcup \cdots \bigcup V(e_i) \subset Y_{i+1}.$$

(2) $Y \subseteq A(e_r, H_r)$.

Thus we obtain the following proposition by Propositions 4.8, 4.3 and Remark 4.3.

PROPOSITION 4.9.

$$A(e_r,H_r)=V(H).$$

COROLLARY 4.4.

$Z_H = \{e_1, \ldots, e_r\}$ is a minimum solution to the (m+1)- edge-connectivity augmentation of a graph H.

PROPOSITION 4.10.

Suppose that $q \geq 3$. Then, for each i, $1 \leq i \leq r-1$, $A(e_i,H_i)$ is an (m+1)-pendant of H_i if and only if q is odd and i=r-1.

PROOF.

Put

$$A = A(e_i,H_i).$$

First suppose that q is odd and i=r-1. Since $|Y-A| = 1$, H_{r-1} has an m-cut $K = K(\{y\}, H_{r-1})$, where $Y-A = \{y\}$. The K-block $B(A,K; H_{r-1})$ contains at least one (m+1)-pendant of H_{r-1} (by Proposition 3.3 of [20-22]). Hence if we assume that A is not an (m+1)-pendant of H_{r-1} then we have an (m+1)-pendant y' of H_{r-1} such that

$$y' \in B(A,K;H_{r-1}) - A,$$

meaning that

$$y' \in Y-A, \quad y' \neq y,$$

a contradiction. Thus A is an (m+1)-pendant of H_{r-1}.

Conversely suppose that A is an (m+1)-pendant of H_i. Then A is external. Put

$$K = K(A, H_i),$$

where $|K| = m$. Let B_A, B be the two K-blocks of H_i such that

$$B_A = A. \quad B = V(H_i) - A$$

Assume that either q is even or q is odd and $i < r-1$. Then there is an edge e_{i+1}, for which

$$V(e_{i+1}) \subseteq B \text{ (by Remark 4.3–(1))}.$$

Hence K is also a (B_A, B)-cut of $H_{i+1} = H_i + e_{i+1}$, meaning that

$$A \bigcap A(e_{i+1}, H_{i+1}) = \phi.$$

This contradicts Proposition 4.8.

Q.E.D.

COROLLARY 4.5.

For each i, $1 \leqslant i \leqslant r$, the pair of vertices of $V(e_i)$ is an admissible pair with respect to H_{i-1}

4.2.3. DETERMINING Z(m+1) FROM Z_H

We describe how to determine Z(m+1) from Z_H obtained by the procedure ***hfind***. For each i, $1 \leqslant i \leqslant r$, put

$$e_i = (v_{i1}, v_{i2}).$$

$$V_i = \bigcup_{v \in A(e_i, H_i)} \rho_{m+1}^{-1}(v)$$

Each $\rho_{m+1}^{-1}(v)$ is an external (m+1)-pendant of G, and Proposition 3.3 of [20-22] shows that there are two sequences

$$S_{ij}(k) \subseteq \cdots \subseteq S_{ij}(m+2) \subseteq \rho_{m+1}^{-1}(v_{ij}), \ j = 1, 2.$$

where $S_{ij}(t')$ denotes an external t'-pendant of G, $t' = m+2, \ldots, k$. For each i, $1 \leqslant i \leqslant r$, choose two vertices

$$w_{ij} \in S_{ij}(k), \ j = 1, 2.$$

and put

$$g_i = (w_{i1}\ w_{i2}),$$

where $g_i \notin E(G)$, $w_{i1} \neq w_{i2}$. Let

$$G_0 = G,\ G_i = G_{i-1} + g_i,\ i=1,\ldots,r.$$

PROPOSITION 4.11.

For each i, $1 \leq i \leq r$, the following (1) - (3) hold:

(1) V_i is an (m+1)-component of G_i, and if S is any (m+1)-component of G_i such that $S \bigcap V_i = \phi$ then $S \in \pi(m+1)$.

(2) $d(V_i,G_i) = d(V_i,G) = d(A(e_i, H_i), H_i) = d(A(e_i,H_i),H)$.

(3) V_i is an external (m+1)-pendant of G_i if and only if $A(e_i, H_i)$ is an external (m+1)-pendant of H_i.

PROOF.

First we prove (1), where it suffices to consider the case where

$$V_i \subset V(G), \text{ or } i<r.$$

Let $S_1,S_2 \in \pi(m+1)$, $S_1 \neq S_2$, and let K_G be any (S_1,S_2)- cut of G, where $|K_G|=m$. There are distinct vertices $u_j \in V(H)$ such that

$$S_j = \rho_{m+1}^{-1}(u_j),\ j=1,2.$$

Proposition 2.1 shows that H has a (u_1, u_2)-cut $K_H = \rho_{m+1}^{-1}(K_G)$, for which

$$B(u_j,K_H;H) = \{\rho_{m+1}(S): S \in \pi(m+1),\ S \subset B(S_j,K_G;G)\},\ j=1,2.$$

Put

$$A = A(e_i,H_i),\ B(u_j) = B(u_j,K_H;H),\ B(S_j) = B(S_j,K_G;G).$$

Suppose that

$$S \subset V_i,\ j=1,2.$$

Then

$$u_j \in A,\ j=1,2.$$

meaning that K_H is no longer a (u_1,u_2)-separator of H_i. There is some $e_t = (v_{t1},v_{t2}) \in Z_H$, $1 \leqslant t \leqslant i$, such that

$$v_{tj} \in B(u_j),\ j=1,2.$$

For the corresponding edge $g_t = (w_{t1},w_{t2})$,

$$w_{tj} \in \rho_{m+1}^{-1}(v_{tj}) \subseteq B(S_j),\ j=1,2.$$

Therefore any (S_1,S_2)-cut of G cannot be an (S_1,S_2)-separator of G_i.

Next suppose that

$$S_1 \subset V_i, S_2 \subset V(G) - V_i.$$

Then

$$u_1 \in A, u_2 \in V(H_i) - A.$$

H_i has an $(A,\{u_2\})$-cut K_H' with $|K_H'|=m$. Since

$$V(e_j) \subset A,\ j=1,\ldots,i.$$

K_H' is also an $(A,\{u_2\})$-cut of H, and $\{u_2\}$ is an $(m+1)$-component of H_i and of H. Proposition 2.1 shows that G has an (S_1,S_2)-cut K_G' with $|K_G'| = |K_H'|$. Since

$$V(g_j) \subset V_i \subset B(S_1,K_G';G),\ j=1,\ldots,i,$$

K_G' is an (V_i,S_2)-cut of G_i, and $S_2 \in (m+1)$. Thus (1) follows.

If

$$g \in K(V_i,G_i),\ e \in K(A,H_i)$$

then

$$g \in K(V_i,G),\ e \in K(A,H),$$

respectively, and (2) follows. Clearly, any (m+1)-pendant of G_i (of H_i) is external, and (3) follows from (2).

Q.E.D.

PROPOSITION 4.12.

For each i, $1 \leq i \leq r$, the pair $w_{i1}, w_{i2} \in V(g_i)$ is an admissible pair with respect to G_{i-1}.

PROOF.

We will prove the proposition by induction on i.

Inductive basis (i=1): $\rho_{m+1}^{-1}(v_{ij})$, j=1,2, are distinct external (m+1)-pendants of G_{i-1}, and, therefore, the pair w_{i1}, w_{i2} satisfies the edge condition for G_{i-1}. If $q \geq 4$ then, by Propositions 4.10, 4.11, V_i is not an (m+1)-pendant of G_i. Hence the pair w_{ij}, w_{i2} is an admissible pair with respect to G_{i-1}.

Inductive hypothesis ($i \geq 2$): For any t, $1 \leq t < i$, the pair w_{t1}, w_{t2} is an admissible pair with respect to G_{t-1}.

Inductive step ($i \geq 2$): If q is odd and i=r then, by Propositions 4.10, 4.11, $\rho_{m+1}^{-1}(v_{i2}) = V_{i-1}$, which is an (m+1)-pendant of G_{i-1}. (Note that we have assumed that $v_{r-1,1} \neq v_{r1}$, $v_{r-1,2} = v_{r2}$ if q is odd). Hence, regardless of q or i, our choice shows that $\rho_{m+1}^{-1}(v_{ij})$, j=1,2, are always distince external (m+1)-pendants of G_{i-1}.

Suppose that G_{i-1} has at least four external (m+1)-pendants. Since

$$\begin{cases} i-1 < r-1 & \text{if } q \text{ is even,} \\ i-1 < r-2 & \text{if } q \text{ is odd.} \end{cases}$$

Proposition 4.10 shows that V_i is not an (m+1)-pendant of G_i. Thus the pair w_{i1}, w_{i2} is an admissible pair with respect to G_{i-1}.

Q.E.D.

COROLLARY 4.6.

We can set

$$Z(m+1) = \{g_1, \ldots, g_r\}.$$

4.2.4. THE PROCEDURE *FIND* AND ITS TIME COMPLEXITY

The procedure *fnd* is a modified version of the procedure *hfnd*: we find edges $g_i \in Z(m+1)$, add them to G^s, and constructs adjacency lists for $(G^s + Z(m+1))^s$ without handling H. In the procedure *hfnd*, the index i, $1 \leq i \leq r-1$, is used, where $r=\lfloor q/2 \rfloor$, $q = |Y|$. In the procedure *fnd*, we search the (m+1)-level for a pair of (m+1)-pendants not yet processed. Concerning vertices, say v or w, and edges, say t=(v,w), appearing in the procedure *hfnd*, we choose vertices a_v, a_w from corresponding (m+1)-pendants of G and maintain adjacency lists for $(G+(a_v,a_w))^s$ if the edge (a_v,a_w) is added to G. Accordingly, for example, determining if $A(f) \bigcap A(f') = \phi$ at line 7 of the procedure *hfnd* is done by finding vertices a_v, a_w, a_v', a_w' from corresponding (m+1)-pendants of G and by computing M_G values by means of adjacency lists for $\bar{G} = (G^s + \{(a_v,a_w),(a_v',a_w')\})^s$. $H_i = H_{i-1} + e_i$ is maintained as adjacency lists for $(G_i)^s = ((G_{i-1})^s + g_i)^s$.

One edge g_i can be found in $O(kn_v + \alpha(G_{i-1}''))$time, where $G_{i-1}'' = G_{i-1} + \{(a_v,a_w),(a_v',a_v')\}$. If we use Dinic's maximum flow algorithm then the total time of the procedure *fnd* is

$$O(rkn_v + \sum_{i=1}^{r}((m+1)n_v(n_v + i+1))),$$

or

$$O(kn_v^2(n_v + n_e)).$$

If we use MKM's maximum flow algorithm then the total time is

$$O(kn_v^3).$$

4.3. THE IMPROVED ALGORITHM

The improved algorithm repeats the following three steps (1) - (3) at most k-1 times:

(1) The procedure *comptree*, which constructs CT(G) for the current graph G.

(2) Computing (i) - (iv) or (i), (iii) (iv) mentioned in 3.3.3 - (1).

(3) **if** ec(G) = 0 **then** the procedure *connect*, which find Z(1) and constructs adjacency lists for $(G^s + Z(1))^s$ **else** the procedure *fnd*, which finds Z(m+1), m=ec(G), and constructs adjacency lists for $(G^s + Z(m+1))^s$.

Let

$$n_i = |E(G_i)|, \ i=0, \ldots, k.$$

Then the time complexity of each step is as follows:

(1) $O(kn_v^3 n_1)$ (Dinic) or $O(kn_v^4)$ (MKM)

(2) $O(k(n_v + n_1))$

(3) The procedure *connect* $O(kn_v^2)$. The procedure *fnd* $O(kn_v^2(n_v + n_e))$ (Dinic) or $O(kn_v^3)$ (MKM).

Since

$$n_i \leq n_e + kn_v,$$

the total time is

$$O(k^2 n_v^3(kn_v + n_e)) \quad \text{(Dinic)}$$

or

$$O(k^2(n_v^4 + kn_v + n_e)) \quad \text{(MKM)}.$$

We note that space complexity is $O(kn_v + n_e)$ plus space required by a maximum flow algorithm.

5. CONCLUDING REMARKS

We have proposed an improved version of an algorithm for finding a minimum solution to the k-edge-connectivity augmentation problem. Taking advantage of the results in [9] to reduce time complexity in constructing component trees, as mentioned at the end of 3.3.2, will lead to a more efficient algorithm. We can also expect that a maximum flow algorithm will spend less time on H than on G. If we actually construct H and use the procedure *hfind* then we may be able to obtain a more efficient algorithm with the increase in space complexity.

ACKNOWLEGEMENT

The author would like to thank Professor Franco P. Preparata, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, for his encouragement and giving the author the opportunity of doing research at his laboratory.

REFERENCES

1 A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).

2 B. Bollobás, Extremal Graph Theory (Academic Press, London, 1978).

3 K.P. Eswaran and R.E. Tarjan, Augmentation problems, *SIAM J. Comput.*, 5, (1976), 653-665.

4. S. Even, Graph Algorithms (Pitman, London, 1979).

5. H. Frank and W. Chou, Connectivity considerations in the design of survivable networks, *IEEE Trans. on Circuit Theory*, CT-17 (1970), 486-490.

6 G. Frederickson and J. Ja'ja, Approximation algorithms for several graph augmentation problems, *SIAM J. Comput.* 10 (1981), 270-283.

7. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness (Freeman, San Francisco, CA, 1978).

8. F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1969).

9. T.C. Hu, Integer Programming and Nerwork Flows (Addison-Wesley, Reading, MA, 1969).

10. Y. Kajitani and S. Ueno, The minimum augmentation of a directed tree to a k-edge-connect directed graph, Technical Research Reports I.E.C.E. Japan, CAS83-3(1983-05), 17-23.

11 D.E. Knuth, The Art of Computer Programming, vol. 1 (Addison-Wesley, Reading, MA, 1969).

12. W. Mader, A reduction method for edge-connectivity in graphs, in: B. Bollobás, ed., Advances in Graph Theory (North-Holland, Amsterdam, 1978), 145-164.

13. V.M. Malhotra, M.P. Kumar and S.N. Maheshwari, An $O(|V|^3)$ algorithm for finding maximum flows in networks, *Information Processing Letters*, 7(1978), 277-278

14. T. Masuzawa, K. Hagihara, K. Wada and N. Tokura, The k-node-connectivity augmentation problem for directed binary trees, *I.E.C.E. Japan Trans. (D)* J67-D, 1(1984) 77-84, (Japanese)

15. A. Rosenthal and A. Goldner, Smallest augmentations to biconnect a graph, *SIAM J. Comput.* 6(1977), 55-66.

16. S. Ueno, Y. Kajitani and H. Wada, The minimum augementation of trees to k-edge-connected graphs, Technical Research Reports I.E.C.E. Japan, IN-83-6(1983-05)1-6, (Japanese)

17. T. Watanabe and A. Nakamura, Vertex-connectivity augmentation problems, Technical Research Reports I.E.C.E. Japan, AL81-26(1981-07) 1-8, (Japanese)

18. T. Watanabe and A. Nakamura, On a smallest augmentation to triconnect a graph, Technical Report No. C-18, Department of Applied Math., Faculty of Engineering, Hiroshima University, Higashi-Hiroshima, Japan (1983-07; revised 1985-11).

19. T. Watanabe, A. Nakamura and M. Takahashi, Augmentation problem for a vertex subset of a graph, Technical Research Reports I.E.C.E. Japan AL83-89(1984-03) 107-114, (Japanese)

20. T. Watanabe and A. Nakamura, K-edge-connectivity augmentation problem, Technical Research Reports I.E.C.E. Japan AL83-90(1984-03) 115-122, (Japanese)

21. T. Watanabe and A. Nakamura, On a smallest augmentation to k-edge-connect a graph Technical Report No. C-20, Department of Applied Math., Faculty of Engineering Hiroshima University, Higashi-Hiroshima, Japan (1984-07; revised 1986-07).

22. T. Watanabe and A. Nakamura, Edge-connectivity augmentation problems, to appear in *J Comput System Sci.*.

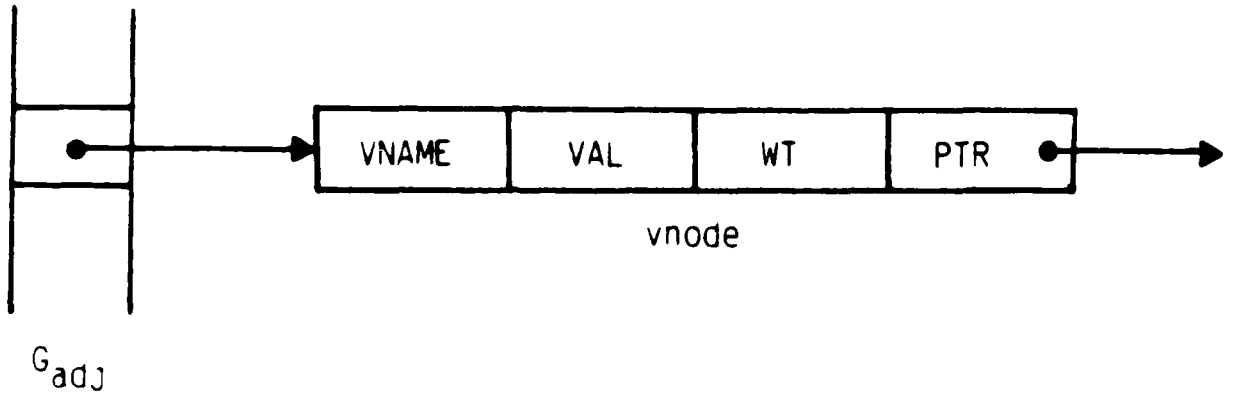


Fig. 1.

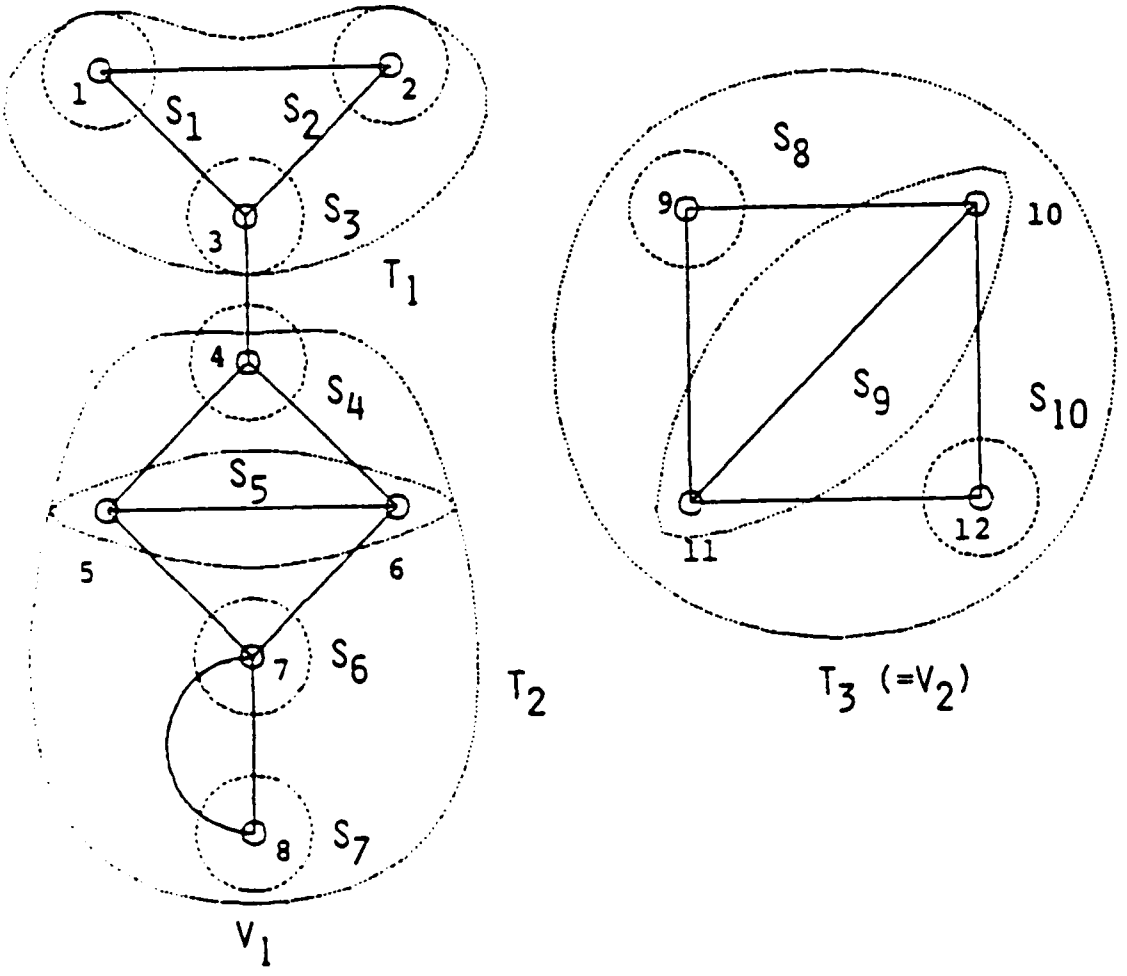


Fig. 2.

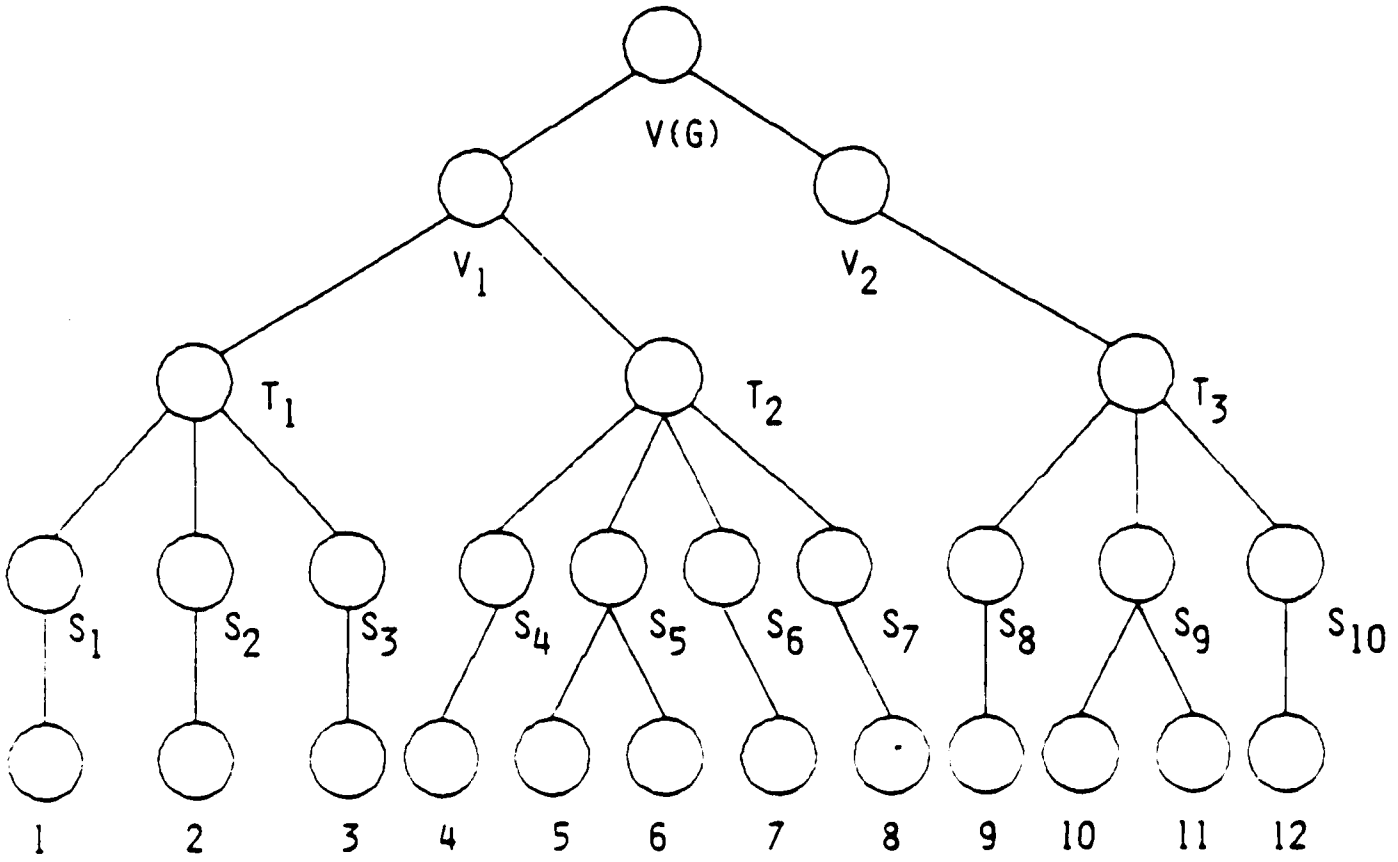


Fig. 3.

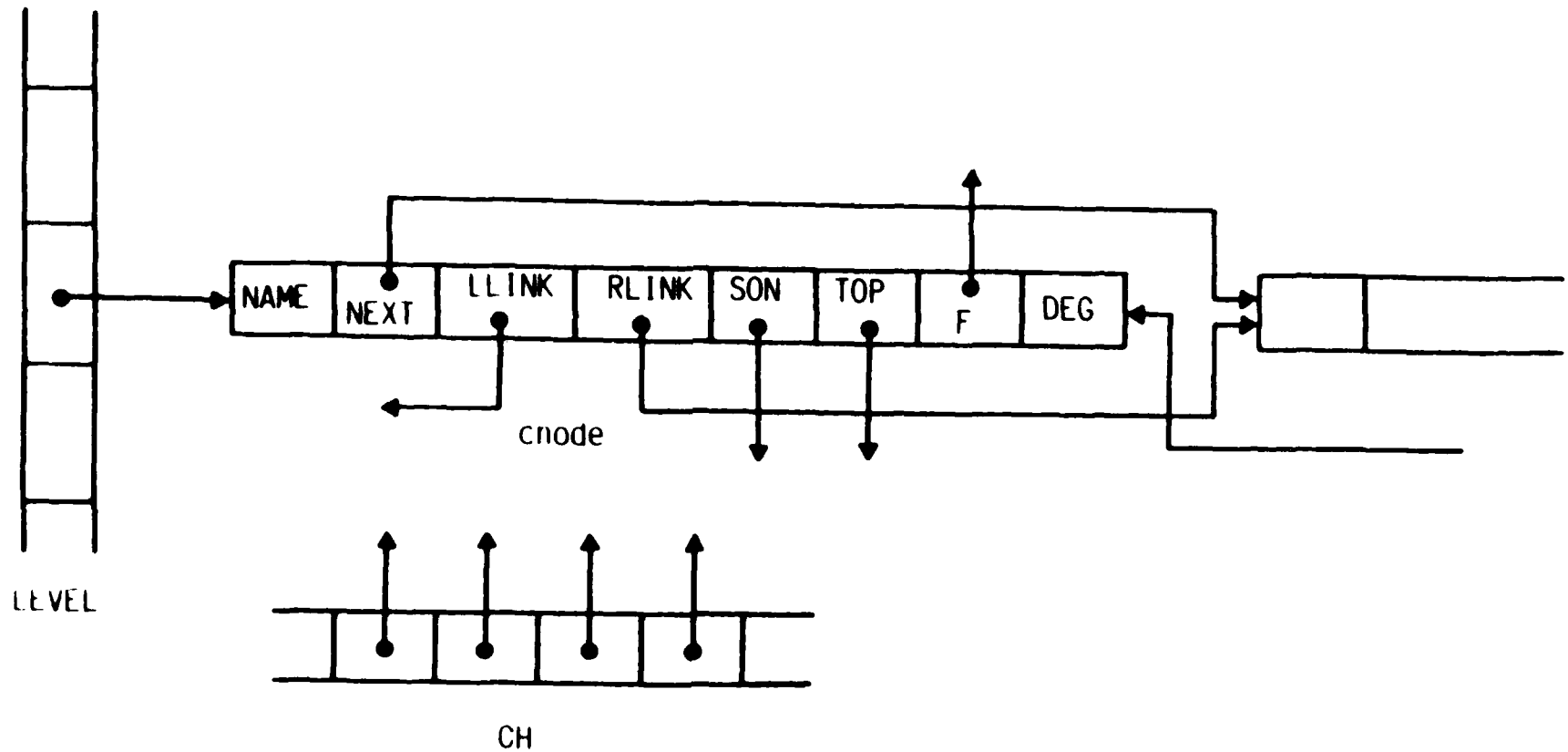


Fig. 4.

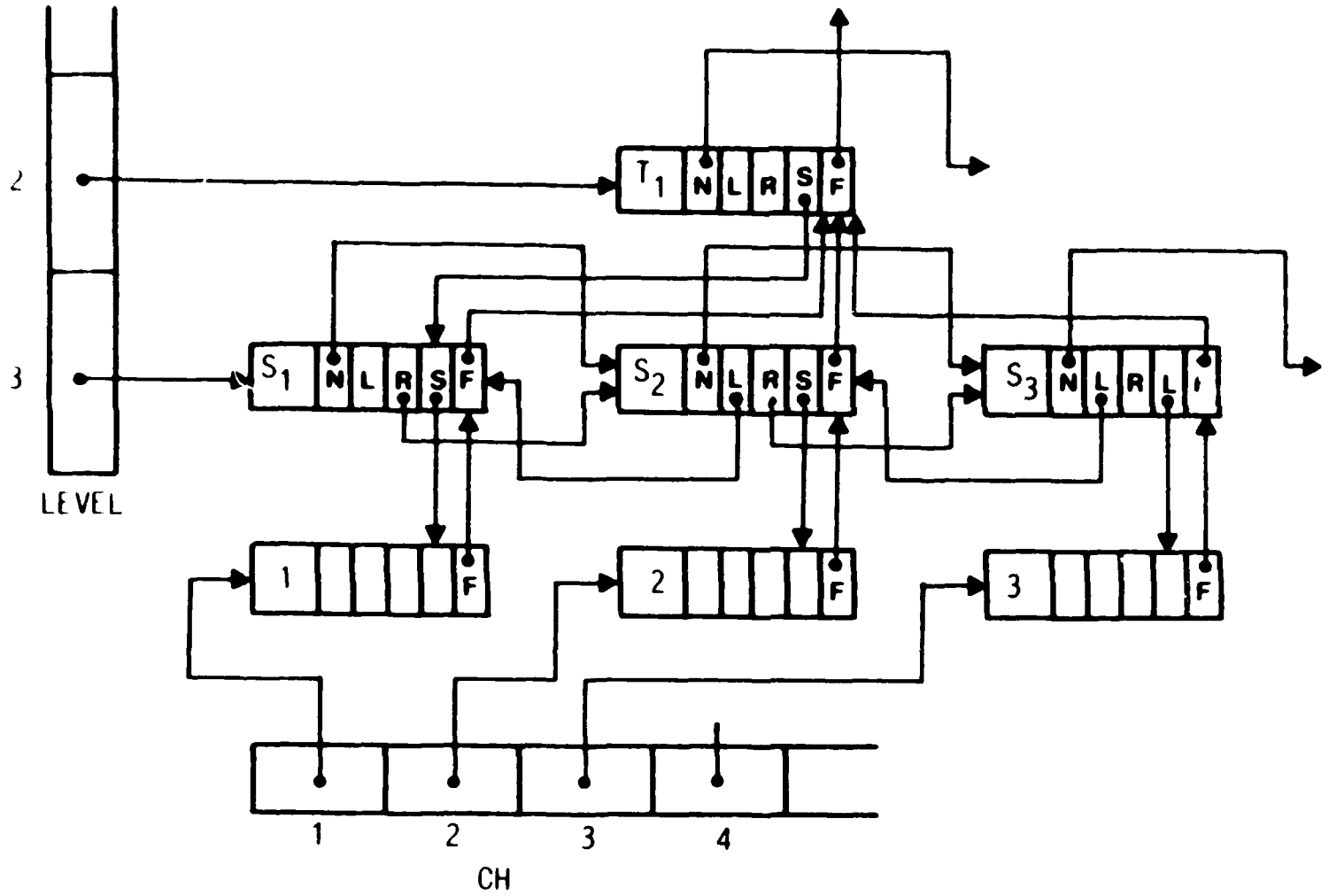


Fig. 5.

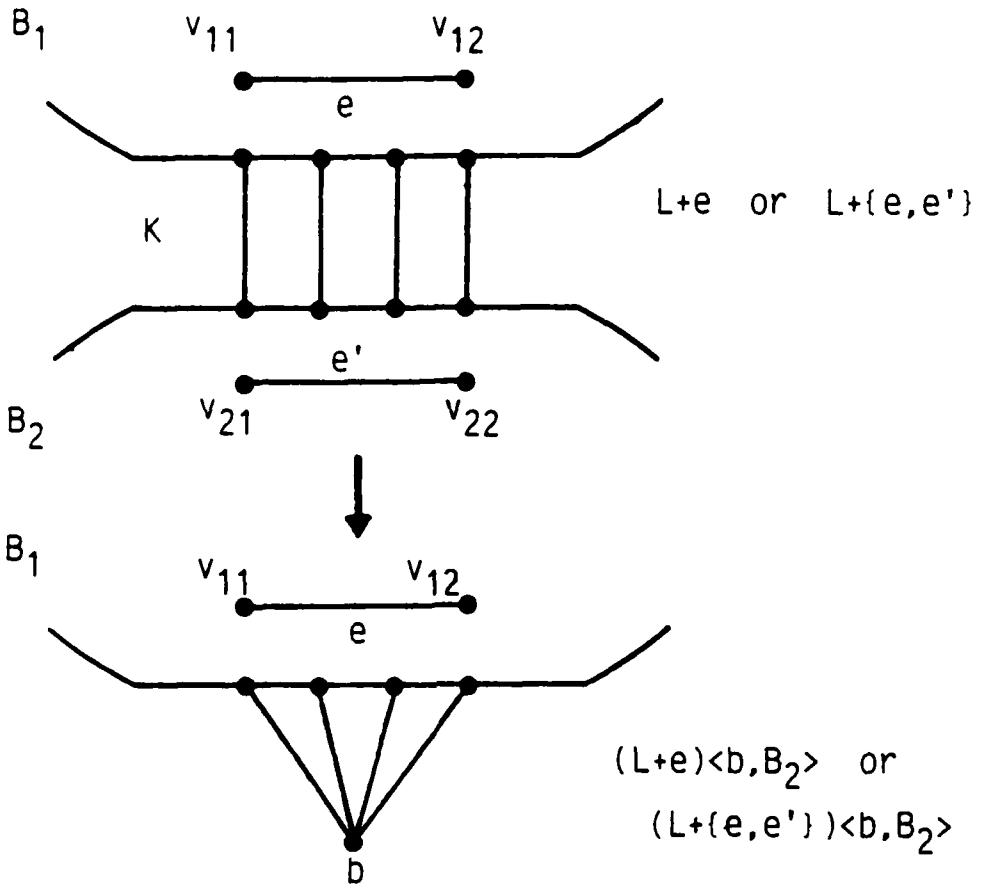


Fig. 6.

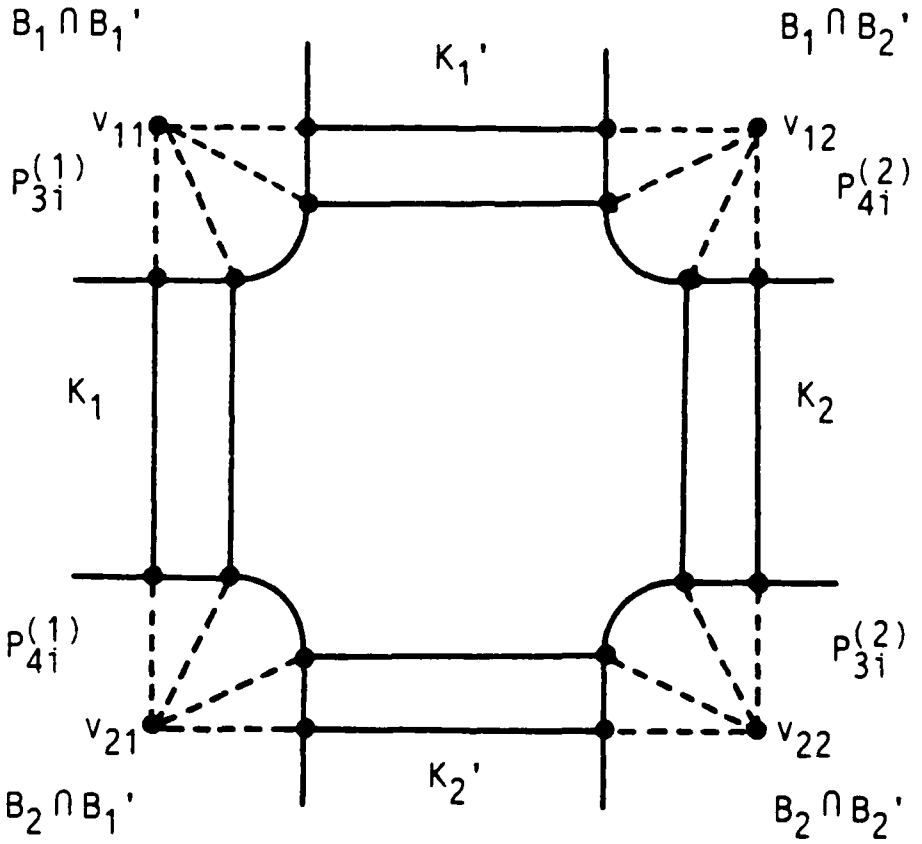

$B_1 \cap B_1'$
K_1'
$B_1 \cap B_2'$
v_{11}
v_{12}
$P_{3i}^{(1)}$
$P_{4i}^{(2)}$
K_1
K_2
$P_{4i}^{(1)}$
$P_{3i}^{(2)}$
v_{21}
v_{22}
K_2'
$B_2 \cap B_1'$
$B_2 \cap B_2'$

Fig. 7.

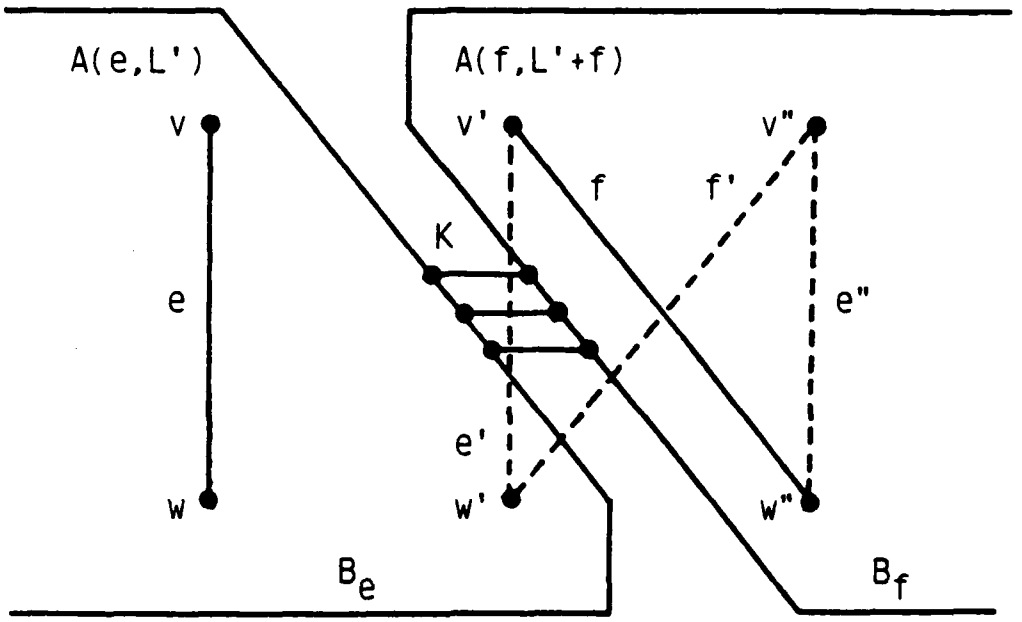


Fig. 8.

END

8-87

DTIC